ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA STROJNÍ

BAKALÁŘSKÁ PRÁCE

2013/2014 Marek Kysela

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta strojní
Akademický rok: **2013/2014**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Marek KYSELA**
Osobní číslo:	**S11B0172P**
Studijní program:	**B2301 Strojní inženýrství**
Studijní obor:	**Průmyslové inženýrství a management**
Název tématu:	**Plošná optimalizace - Bin Packing Problem**
Zadávající katedra:	**Katedra průmyslového inženýrství a managementu**

Zásady pro vypracování:

1. Úvod do řešené problematiky
2. Analýza současného stavu
3. Analýza použitelných metod
4. Uplatnění metod
5. Závěr

-

Rozsah grafických prací: **0 výkresů**

Rozsah pracovní zprávy: **30 - 40 stran**

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

1. **Kotásek, J.** *Plánování nakládky jako součást aplikace DCIx.* **Diplomová práce, Plzeň: ZČU-KIV, 2013.**
2. **Teda, J.** *Nové možnosti informačních systémů s využitím inteligentních modulů.* **Ostrava: Vítkovice ITS a. s., 2006.**
3. **Lodi, A.** *Algorithms for Two-Dimensional Bin Packing and Assignment Problems.* **Padova: Universita Delgi Di Bologna, 1999.**
4. **Jylänki, J. http://clb.demon.fi. 27. 2. 2010. http://clb.demon.fi/files/RectangleBinPack.pdf**
5. **ULRYCH, Z., RAŠKA, P.** *VYZTYMDP: Modelování a simulace a DP, e-book.* **Plzeň: ZČU-KPV, 2012. ISBN 978-80-87539-15-6.**
6. **ULRYCH, Z., VOTAVA, V., RAŠKA, P., HOŘEJŠÍ, P.** *ŽIVDIG: Simulace výrobních systémů a procesů, e-book.* **Plzeň: ZČU-KPV, 2013. ISBN 978-80-87539-37-8.**

Vedoucí bakalářské práce: **Ing. Pavel Raška, Ph.D.**
Katedra průmyslového inženýrství a managementu

Konzultant bakalářské práce: **Doc. Ing. Zdeněk Ulrych, Ph.D.**
Katedra průmyslového inženýrství a managementu

Datum zadání bakalářské práce: **23. září 2013**

Termín odevzdání bakalářské práce: **27. června 2014**

Doc. Ing. Jiří Staněk, CSc.
děkan

L.S.

Doc. Ing. Michal Šimon, Ph.D.
vedoucí katedry

V Plzni dne 23. září 2013

Prohlášení o autorství

Předkládám tímto k posouzení a obhajobě bakalářskou/diplomovou práci, zpracovanou na závěr studia na Fakultě strojní Západočeské univerzity v Plzni.
Prohlašuji, že jsem tuto bakalářskou/diplomovou práci vypracoval samostatně, s použitím odborné literatury a pramenů, uvedených v seznamu, který je součástí této bakalářské/diplomové práce.

V Plzni dne: ……. **. . . . . .**
podpis autora

Poděkování

Chtěl bych poděkovat všem z Katedry průmyslového inženýrství a managementu za jejich trpělivost, pomoc a užitečné rady nezbytné k dokončení mé práce. Dále své rodině za podporu ve studiu a také bohům za příležitosti, které mi byly v životě nabídnuty.

ANOTAČNÍ LIST BAKALÁŘSKÉ PRÁCE

AUTOR	**Příjmení** Kysela	**Jméno** Marek
STUDIJNÍ OBOR	B2301 Průmyslové inženýrství a management	
VEDOUCÍ PRÁCE	**Příjmení (včetně titulů)** Ing. Raška, Ph.D.	**Jméno** Pavel
PRACOVIŠTĚ	ZČU - FST - KPV	

DRUH PRÁCE	~~**DIPLOMOVÁ**~~	**BAKALÁŘSKÁ**	**Nehodící se škrtněte**
NÁZEV PRÁCE	Bin Packing Problem		

FAKULTA	strojní		**KATEDRA**	KPV		**ROK ODEVZD.**	2014

POČET STRAN (A4 a ekvivalentů A4)

CELKEM	61		**TEXTOVÁ ČÁST**	46		**GRAFICKÁ ČÁST**	15

STRUČNÝ POPIS (MAX 10 ŘÁDEK) **ZAMĚŘENÍ, TÉMA, CÍL POZNATKY A PŘÍNOSY**	Tato bakalářská práce se zabývá problematikou dvojdimenzionálního obdélníkového Bin Packing Problemu. Ten obecně slouží k minimalizaci nevyužitého místa. Nejčastěji slouží k optimalizaci uložení zboží v kontejneru. Dále slouží k optimalizaci využití místa při skladování, k minimalizaci odpadu při řezání plechů, skla a dalších materiálů. Práce je zaměřena na policové algoritmy pro 2D BPP.
KLÍČOVÁ SLOVA	Bin Packing Problem, optimalizace, logistika, policové algoritmy, pascal

SUMMARY OF BACHELOR SHEET

AUTHOR	**Surname** Kysela	**Name** Marek
FIELD OF STUDY	B2301 Industrial Engineering and Management	
SUPERVISOR	**Surname (Inclusive of Degrees)** Ing. Raška, Ph.D.	**Name** Pavel
INSTITUTION	ZČU - FST - KPV	
TYPE OF WORK	~~**DIPLOMA**~~	**BACHELOR**	**Delete when not applicable**
TITLE OF THE WORK	Bin Packing Problem	

FACULTY	Mechanical Engineering		**DEPARTMENT**	Machine Design		**SUBMITTED IN**	2014

NUMBER OF PAGES (A4 and eq. A4)

TOTALLY	61		**TEXT PART**	46		**GRAPHICAL PART**	15

BRIEF DESCRIPTION **TOPIC, GOAL, RESULTS AND CONTRIBUTIONS**	This thesis deals with rectangular two-dimensional Bin Packing Problem. It is generally used to minimize wasted space. It is mostly used to optimize the storage of goods in the container. It is also used to optimize the use of space for storage, to minimize waste when cutting sheet metal, glass and other materials. The work is focused on shelf algorithms for 2D BPP.
KEY WORDS	Bin Packing Problem, optimization, logistics, shelf algorithms, pascal

Obsah

1 Úvod

Bin Packing Problem (dále jen BPP) slouží obecně k minimalizaci nevyužitého místa. Jeho řešení má mnohá použití. Nejčastěji slouží k optimalizaci uložení zboží v kontejneru a v následném uložení těchto kontejnerů v nákladním prostoru dopravních prostředků, ať už se jedná o leteckou, námořní, železniční nebo silniční dopravu. Následně slouží k optimalizaci využití místa při skladování, k minimalizaci odpadu při řezání plechů, skla a dalších materiálů. Mezi další jeho využití patří optimalizace rozložení nářadí a součástí v montážním průmyslu, ale také třeba rozmístnění reklam v novinách.

BPP patří ke standardním optimalizačním problémům. Je to kombinatorický NP-těžký problém, kde je množina **předmětů** o různých velikostech uložena do co nejmenšího počtu homogenních **kontejnerů** (binů). Původní definice se vztahuje pouze na jednorozměrné kontejnery a předměty (1), následně vznikla spousta dalších odvozených verzí BPP zabývajících se řešením vícerozměrných problémů. (2)

Přesné vstupy a cíl se liší podle použití. Například množina předmětů může být tak velká, že je považována za nekonečnou, a množina kontejnerů je považována za konečnou. Výstupem potom je počet předmětů, který lze uložit do daného počtu kontejnerů. Nebo naopak je dán omezený počet předmětů, které mají být uloženy, a neomezený počet kontejnerů, jde tedy pouze o to, aby bylo kontejnerů využito co nejméně. (3)

Dále se může BPP dělit na online BPP a offline BPP. U **online** BPP přichází ze vstupu jeden předmět za druhým a příchozí předmět musí být ihned uložen do jednoho z kontejnerů a teprve potom je obdržen následující předmět, u kterého dopředu nejsou známy jeho rozměry. Opakem této varianty je **offline** BPP, u kterého je dopředu známa celá množina předmětů k uložení do kontejnerů, včetně jejich rozměrů. (4)

Tato práce je zaměřena na online dvojdimenzionální obdélníkový BPP.

1.1 Vstupy

Jako vstupy pro řešení BPP jsou definovány množina předmětů a množina homogenních kontejnerů.

1.1.1 Předměty

Je dána množina obdélníkových předmětů R_i o určité šířce w_i a výšce h_i. Při třídimenzionálním BPP přibývá ještě třetí rozměr a to hloubka d_i.

Pro předměty platí následující omezení:

Předměty se nesmí částečně nebo úplně překrývat a ani nemůže být jeden vložen do druhého. (3)

Rozměry předmětů nesmí přesáhnout rozměry kontejneru a předměty nelze dělit, musí být uloženy do jednoho kontejneru jako celek. (3)

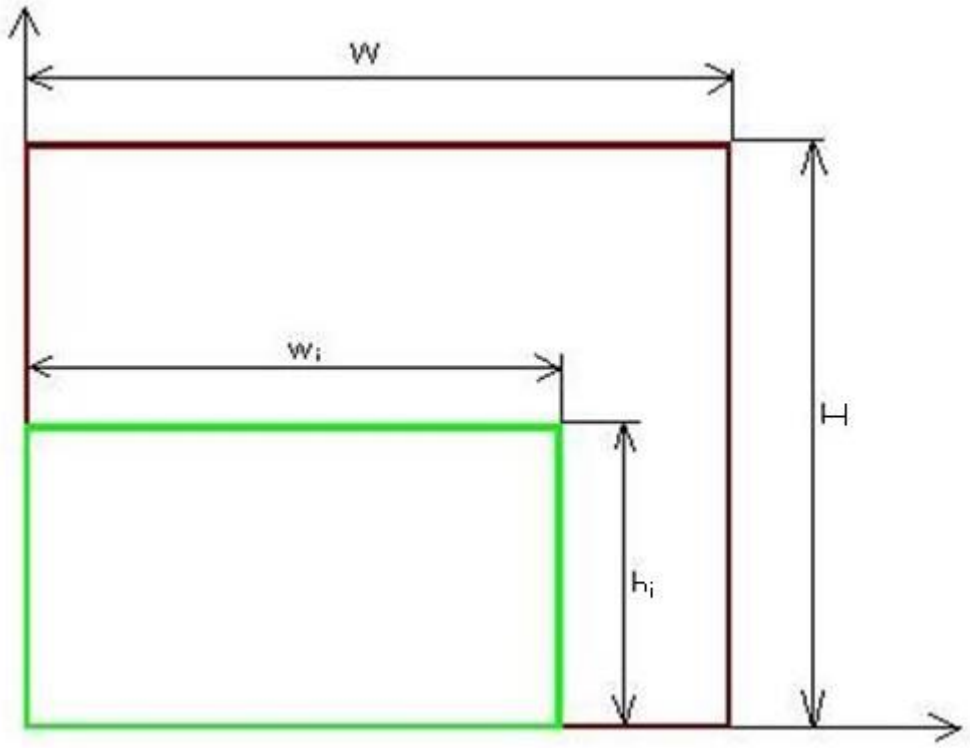
Předměty mohou být do kontejneru uloženy pouze tak, že jsou jejich hrany rovnoběžně s hranami kontejneru, předměty mohou být otočeny pouze o 90°. (4)

1.1.2 Kontejnery

Je dána množina homogenních obdélníkových kontejnerů. Všechny kontejnery mají stejné rozměry: šířku W_b, výšku H_b a u třídimenzionálního BPP ještě hloubku D_b.

Při řešení reálného problému může být nadefinován maximální počet kontejnerů. (3)

Podle použitého algoritmu může být najednou buď otevřen pouze jeden kontejner nebo může být zároveň otevřeno více kontejnerů a algoritmus mezi nimi vybírá, do kterého vloží další předmět. Daný algoritmus pak určuje maximální počet kontejnerů, které mohou být otevřeny zároveň. K otevření dalšího kontejneru je nutné nějaký z otevřených kontejnerů zavřít. (4)



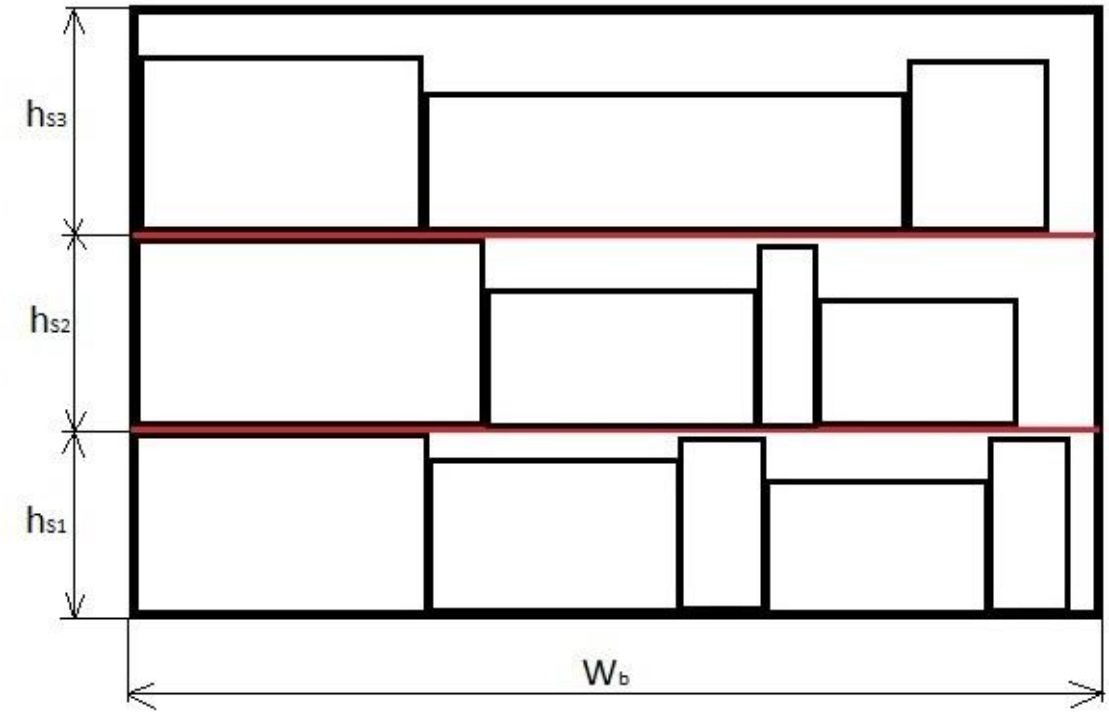
Obr. 1-1 Předmět uložený v kontejneru

2 Algoritmy pro dvojdimenzionální obdélníkový BPP

V této kapitole budou popsány algoritmy řešící dvojdimenzionální obdélníkový BPP. Algoritmy jsou seřazeny ve skupinách podle zásadních datových struktur, které charakterizují proces ukládání předmětů do kontejnerů a k zobrazení zbylého volného místa v kontejneru. (5)

2.1 Policové (shelf) algoritmy

Policové algoritmy (nebo takové levelové algoritmy) patří mezi základní algoritmy, které se dají použit pro proces ukládání předmětů. Police je definována jako obdélníková část kontejneru o šířce W_b a výšce h_s. Jednotlivé police jsou v kontejneru uspořádány odspoda nahoru. Do polic jsou předměty skládány zleva doprava. Poslední police, tedy police úplně nahoře, se nazývá **otevřená** police. Protože jsou předměty do polic skládány odspoda nahoru, místo nad otevřenou policí je vždy nevyužito. To umožňuje nastavení výšky této police, i když jsou v ní už naskládány předměty. Pro ostatní police pod otevřenou policí toto neplatí, výšku těchto polic měnit nemůžeme, a proto se těmto policím říká **uzavřené**.



Obr. 2-1 Uložení předmětů pomocí policového algoritmu. (4)

Pokud je předmět o rozměrech w a h ukládán do police o rozměrech W_b a h_s je potřeba zvolit, jestli bude předmět otočen nebo jestli zůstane tak, jak do algoritmu vstupuje. Tedy jestli bude předmět uložen na výšku (min(w,h), max(w,h)) nebo na šířku (max(w,h), min(w,h)). Vlastní volba proběhne u všech algoritmů takto:

1) Pokud se jedná o první předmět na nové polici, bude předmět uložen na šířku, aby se minimalizovala výška této nové police.

2) Pokud lze předmět uložit na výšku ($max(w,h) < h_s$), uložíme ho tak. Tímto je docíleno minimalizace prázdného místa mezi vrchní stranou předmětu a stropem police.

3) Pokud nelze předmět uložit na výšku, ale lze ho uložit na šířku, je tak uložen.

Obrázek 2-1 zobrazuje jednoduché ukládání předmětů pomocí policového algoritmu. Červené čáry zobrazují stropy (výšky) jednotlivých polic.

2.1.1 Shelf Next Fit (SHELF-NF)

Shelf Next Fit je úplně nejjednodušší algoritmus k realizaci ukládání předmětů. Speciální vlastností tohoto algoritmu je ta, že vyžaduje pouze konstantní množství pracovní paměti. Všechny ostatní algoritmy používají datovou strukturu, která je minimálně lineárně závislá na počtu předmětů, které jsou již uloženy. Výsledky tohoto algoritmu jsou bohužel hodně vzdáleny od výsledků optimálního uložení.

Algoritmus se pokusí předmět uložit na výšku (pokud se nejedná o úplně první předmět, ten je uložen vždy na šířku), pokud nelze předmět takto uložit, otočí ho a pokusí se ho uložit na šířku. Pokud nelze ani to, zavírá danou polici a otevírá novou. Pokud není dostatek místa pro novou polici (s předmětem uloženým na šířku) algoritmus končí.

2.1.2 Shelf First Fit (SHELF-FF)

Zapomínat na volné místo v uzavřených policích po otevření nové je velmi nehospodárné. Proto si všechny SHELF-NF algoritmy udržují seznam dříve zavřených polic, aby do nich i nadále mohly být ukládány předměty, pokud je to možné. Pokud existuje více polic, do kterých by předmět mohl být uložen, je potřeba jednu z nich zvolit. Pro tuto volbu existuje několik variant. U metody SHELF-FF je předmět vždy ukládán do první (nejspodnější) police, do které předmět může být uložen. U této metody předmět, který lze uložit do některé z uzavřených polic, šetří místo na polici otevřené. Oproti SHELF-NF může tedy tento algoritmus ušetřit nějaké místo, pokud existuje možnost uložení jednoho nebo více předmětů do některé z dříve uzavřených polic.

2.1.3 Shelf Best Width Fit (SHELF-BWF)

Tato metoda vychází z metody SHELF-FF. Algoritmus nejprve projde všechny police, a zjistí, do kterých by mohl být předmět uložen. U tohoto algoritmu bude z těchto polic následně vybrána ta police, ve které bude po uložení předmětu šířka zbývajícího volného místa nejmenší.

2.1.4 Shelf Best Height Fit (SHELF-BHF)

Protože hrany oddělující police jsou přímé čáry, při uložení předmětu o menší výšce, než je výška police, vzniká pruh nevyužitého místa mezi horní hranou předmětu a stropem police. Algoritmus nejprve projde všechny police, a zjistí, do kterých by mohl být předmět uložen. Pro minimalizaci zmíněného nevyužitého místa uloží algoritmus předmět do takové police, kde bude rozdíl výšek h_s-h minimální.

2.1.5 Shelf Best Area Fit (SHELF-BAF)

U SHELF-BHF algoritmu může nastat situace, kdy bude shodný rozdíl výšek h_s-h při uložení předmětu na výšku a na šířku (v různých policích). Pokud tato situace nastane, vynásobí se tento rozdíl aktuální šířkou předmětu. Tento součin je roven ploše mezi horní hranou předmětu a stropem police. Předmět bude uložen tak, aby byla tato plocha minimální.

2.1.6 Shelf Worst Width Fit (SHELF-WWF)

Zatímco SHELF-BWF se snaží zaplnit šířku každé police, co nejvíce je to možné. SHELF-WWF se snaží o úplný opak. Snaží se na každé polici udržet co největší možnou šířku nevyužitého místa. Algoritmy SHELF-BWF a SHELF-WWF jsou si úplnými opaky, přesto není možné o jednom tvrdit, že by jeho uložení předmětů bylo lepší než u toho druhého. U algoritmu SHELF-WWF je navíc přijato pravidlo, že pokud je ukládán předmět o šířce w a je nalezena police, ve které zbývá právě šířka w volného místa, je okamžitě vybrána tato police a předmět je do ní uložen.

Podle stejného vzoru lze definovat i algoritmy Shelf Worst Height Fit a Shelf Worst Area Fit. Ale protože policové algoritmy nevyužívají místo mezi horní stranou každého předmětu a stropem police, zvyšování tohoto rozdílu by vedlo k maximalizaci nevyužitého místa, a tím pádem jsou tyto metody suboptimální. Jedinou výjimkou by byla kombinace s následujícími metodami.

2.1.7 Shelf Floor-Ceiling

Všechny výše popsané algoritmy mají stejný problém. Neumějí využít volné místo, které vzniká, když se výška předmětu nerovná výšce police. K tomu slouží varianta Shelf Floor-Ceiling (6), kde jsou předměty na vstupu zprvu seřazeny sestupně podle rozměru delší strany. Předměty jsou skládány do polic postupně zleva doprava podél dna police. Po uzavření police a tedy i po nastavení výšky police se předměty začnou ukládat zprava doleva podél stropu police. Následně se otevře další police a postup se opakuje.

2.1.8 The Waste Map Improvement

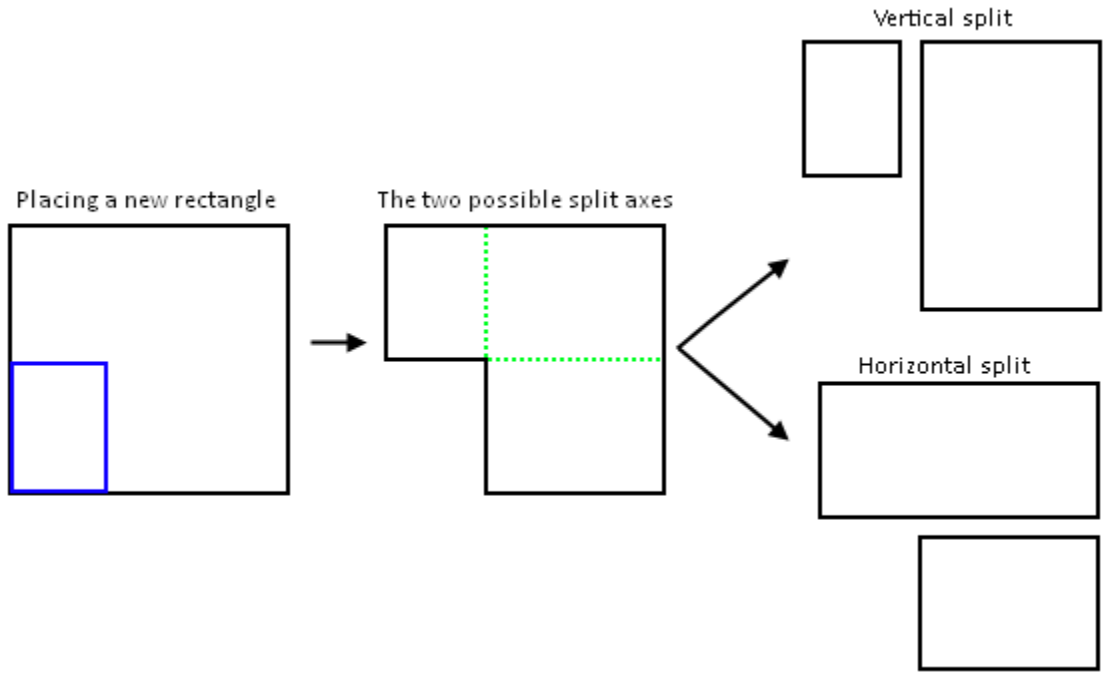
Další metodou, která se snaží využít nadměrné množství volného místa u policových algoritmů, je metoda nazývaná Waste Map. Jelikož je gilotinový algoritmus jednoduchou a efektivní cestou ke skladování nevyužitých míst kontejneru, je tohoto algoritmu využito ke sledování všech míst, která by se jinak nevyužila.

Pro policový algoritmus je postup následovný. Předměty se začnou ukládat zahájením policového algoritmu a zahájením substrukturního případu gilotinového ukládacího algoritmu.

Kdykoliv je uzavřena police, najdou se všechny rozdělené obdélníky volného místa. Tyto obdélníky volného místa uložíme do datové struktury gilotinového algoritmu, který byl zpočátku nulový. Když se začne ukládat další předmět, je nejdříve odzkoušeno, jestli může být předmět uložen gilotinovým algoritmem do již uzavřené police, pokud ne, uložíme ho policovým algoritmem.

2.2 Gilotinové algoritmy

V této kapitole bude k problému ukládání předmětů do kontejneru přistupováno zcela rozdílně. GIlotinové algoritmy jsou založené na operaci, která bude nazývána **rozložení gilotinového rozdělení**. Což je procedura umisťování předmětu do rohu volného obdélníku v kontejneru, po kterém je zbylé volné místo ve tvaru L znovu rozděleno na dva samostatné obdélníky. Tato procedura je znázorněna na obrázku Obr. 2-2.

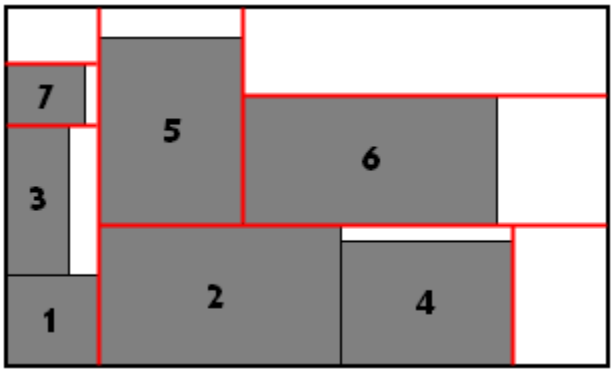


Obr. 2-2 Rozdělení volného místa pomocí gilotinového algoritmu. Po uložení předmětu do kontejneru jsou dvě možnosti rozdělení zbývajícího volného místa. (4)

Gilotinový algoritmus funguje následovně. Je zachovávána množina obdélníků $F = \{F_1,...,F_n\}$, která zastupuje volné místo v kontejneru. Tyto obdélníky se nepřekrývají, tudíž pro všechny platí, že průnik libovolných dvou těchto obdélníků je nula. Algoritmus začíná s jediným obdélníkem $F = \{F_1=(W,H)\}$, při každém kroku ukládání předmětů se vezme prvotně volný obdélník F_i, do kterého je vložen následující předmět R=(w,h). Předmět R je vložen do levého spodního rohu obdélníku F_i, který je potom rozdělen pomocí pravidla gilotinového dělení na dva menší obdélníky volného místa F‘ a F‘‘, které nahradí obdélník F_i v seznamu volných obdélníků. Tato procedura pokračuje, dokud se do žádného z volných obdélníků nevejde následující předmět a poté proces pokračuje s novým kontejnerem.

Velkou výhodou těchto algoritmů je jejich přesné sledování volného místa v kontejneru. Na rozdíl od policových algoritmů nikdy neopomenou žádné volné místo v kontejneru. Nedostatkem těchto algoritmů je, že se snaží předmět R vždy uložit pouze do jednoho obdélníku volného místa F_i, dovnitř kterého předmět nejlépe pasuje. Algoritmus se nikdy nepokusí uložit předmět do takové pozice, při které by ležel přes čáru nějakého předchozího gilotinového rozdělení.

Jednoduchý příklad rozložení pomocí gilotinového algoritmu je znázorněn na obrázku 2-3. Červené čáry znázorňují čáry rozdělení, které byly použity pro rozdělení volného místa tak, aby mohlo být zastoupeno množinou F nepřekrývajících se obdélníků. V tomto obrázku se jedná o 8 obdélníků odpovídajících bílému místu na obrázku.



Obr. 2-3 Jednoduchý příklad uložení předmětů pomocí gilotinového algoritmu. Červené čáry znázorňují výběr rozdělení volného místa po uložení předmětu. (4)

Pro dokončení algoritmu je potřeba ještě definovat dvě pravidla. Zaprvé je potřeba určit pravidlo, podle kterého je vybrán obdélník volného místa F_i, do kterého je předmět uložen. Zadruhé je potřeba zvolit, která ze dvou možností rozdělení zbývajícího volného místa bude použita. Vznikne tedy několik variant, jak daný předmět uložit. Dopředu není jisté, která z variant bude lepší než ostatní, proto je lepší na daný problém aplikovat všechny varianty a poté vybrat tu s nejlepším výsledkem.

2.2.1 Guillotine Best Area Fit (GUILLOTINE-BAF)

GUILLOTINE-BAF algoritmus je velmi podobný SHELF-BAF algoritmu. Algoritmus vybírá nejmenší možný obdélník volného místa F_i, do kterého se následující předmět vejde a následně do něj předmět uloží. Tento algoritmus je přirozeným pravidlem pro minimalizaci úzkých pruhů nevyužitého místa.

2.2.2 Guillotine Best Short Side Fit (GUILLOTINE-BSSF)

Pokud je předmět R=(w,h) ukládán do obdélníku volného místa $F_i=(w_f,h_f)$, je možno vzít v úvahu rozdíly v délkách stran těchto dvou obdélníků. GUILLOTINE-BSSF algoritmus se řídí pravidlem výběru takového obdélníku volného místa F_i, u kterého bude po uložení předmětu platit, že minimum z rozdílů w_f-w a h-h_f bude to nejmenší možné. Jinými slovy tento algoritmus minimalizuje velikost menšího rozdílu délek stran.

2.2.3 Guillotine Best Long Side Fit (Guillotine-BLSF)

GUILLOTINE-BLSF algoritmus je podobný algoritmu GUILLOTINE-BSSF. Na rozdíl od něj, se snaží minimalizovat maximum z rozdílů w_f-w a h-h_f. Jinými slovy tento algoritmus minimalizuje velikost většího rozdílu délek stran.

2.2.4 Guillotine Worst Fit Rules

Guillotine Worst Fit algoritmy jsou analogické k výše zmíněným Best Fit algoritmům.

Guillotine Worst Area Fit (GUILLOTINE-WAT) ukládá předmět R do obdélníku volného místa F_i tak, aby zbývající volné místo bylo největší. Platí zde, jako u všech gilotinových algoritmů, speciální pravidlo, že pokud je nalezen obdélník volného místa F_i, který je stejně velký jako ukládaný předmět R, předmět je do něj uložen, protože se jedná o perfektní shodu.

Guillotine Worst Short Side Fit (GUILLOTINE-WSSF) maximalizuje velikost menšího rozdílu v délkách stran.

Guillotine Worst Long Side Fit (GUILLOTINE-WLSF) maximalizuje velikost většího rozdílu v délkách stran.

Snahou těchto algoritmů je ponechat co nejvíce volného místa mezi jednotlivými předměty, co nejdéle je to možné. Snaží se předejít velmi malým a tím pádem nepoužitelným pruhům volného místa.

2.2.5 The Rectangle Merge Improvement (-RM)

Největším problémem gilotinových algoritmů je, že předmět nemůže být libovolně uložen do volného prostoru, kde by byl předmět uložen tak, že by ho protínala čára již provedeného gilotinového rozdělení. Pokud je volné místo již hodně fragmentováno na jednotlivé obdélníky volného místa, algoritmus může nesprávně nahlásit, že pro daný předmět už v kontejneru není volné místo i přesto, že je. Proto je zde předpoklad, že by ukládání dalších předmětů bylo jednodušší, pokud by bylo možné minimalizovat počet čar rozdělujících zbývající volný prostor.

Existuje procedura nazývaná **obdélníkové slučovací zlepšení**. Po každém uložení předmětu do kontejneru, se projdou postupně všechny obdélníky volného místa a hledají se dva sousedící obdélníky volného místa F_i a F_j takové, aby jejich sjednocení mohlo být zastoupeno jedním větším obdélníkem. Pokud jsou takové dva sousedící obdélníky nalezeny, jsou sloučeny do jednoho, který efektivně odstraní rozdělení volného prostoru způsobené čárou předchozího rozdělení, která existovala mezi obdélníky F_i a F_j.

2.3 Pravidla dělení u gilotinových algoritmů

Jelikož dělící osy určují velikost obdélníků volného místa a protože předmět po uložení nemůže překrývat čáru gilotinového dělení, je důležité dávat pozor na to, jak je samotné rozdělení zbývajícího místa provedeno. V této kapitole je popsáno několik různých metod pro výběr, zda provést dělení vertikálně nebo horizontálně.

Je definován obdélník volného místa $F_i=(w_f,h_f)$, do kterého byl právě uložen předmět $R=(w,h)$.

2.3.1 Shorter/Longer Axis Split Rule (-SAS, -LAS)

U nejjednoduššího pravidla je možno stanovit nezávislost dělení na rozměru ukládaného předmětu R a jednoduše rozdělit volné místo horizontálně, pokud platí $w_f < h_f$, nebo vertikálně, pokud platí opak. Toto pravidlo se nazývá Shorter Axis Split Rule (-SAS), tedy pravidlo dělení podle kratší osy.

Opakem tohoto pravidla je pravidlo Longer Axis Split Rule (-LAS), tedy pravidlo dělení podle delší osy. Volné místo je děleno horizontálně, pokud platí $w_f \geq h_f$, nebo vertikálně, pokud platí opak.

2.3.2 Shorter/Longer Leftover Axis Split Rule (-SLAS, -LLAS)

U tohoto pravidla jsou porovnávány zbytkové délky (rozdíly) $w_f - w$ a $h_f - h$ u obdélníku volného místa. U pravidla Shorter Leftover Axis Split Rule (-SLAS), tedy u pravidla dělení podle menšího rozdílu délek, je volné místo děleno horizontálně, pokud platí $w_f - w < h_f - h$, nebo vertikálně, pokud platí opak.

Opakem tohoto pravidla je pravidlo Longer Leftover Axis Split Rule (-LLAS), tedy pravidlo dělení podle většího rozdílu délek. Volné místo je děleno horizontálně, pokud platí $w_f - w \geq h_f - f$, nebo vertikálně, pokud platí opak.

2.3.3 Max/Min Area Split Rule (-MAXAS, -MINAS)

Místo měření a porovnávání délek stran předmětů a obdélníků volného místa je toto pravidlo zaměřeno na porovnání obsahů volného nevyužitého místa.

2.4 Algoritmy maximálních obdélníků

Gilotinové algoritmy popsané v předcházející kapitole jsou velkým zlepšením oproti policovým algoritmům, ale ohraničení dělicími čárami stále způsobuje problémy s praktickým využitím.

Všem těmto nedostatkům lze předejít použitím algoritmů maximálních obdélníků. Tyto algoritmy jsou v určitém smyslu založeny na rozšíření pravidel dělení u gilotinových algoritmů. Tak jako gilotinový algoritmus si algoritmus maximálních obdélníků ukládá seznam volných obdélníků, který reprezentuje volné místo v kontejneru. Na rozdíl od gilotinového algoritmu, který si volí jednu dělící osu ze dvou, algoritmus maximálních obdélníků provádí operaci, která v podstatě odpovídá výběru obou dělících os najednou.

Tento dělící proces je zobrazen v obrázku 2-4. Po uložení předmětu R do levého spodního rohu volného obdélníku F_i dostaneme dva obdélníky prázdného místa F_1 a F_2, které pokrývají volný prostor ve tvaru L vzniklý rozdílem $F_i - R$ a aktualizujeme množinu F a to tak, že z ní odečteme velikost obdélníku F_i a přičteme do ní velikost obdélníků F_1 a F_2.

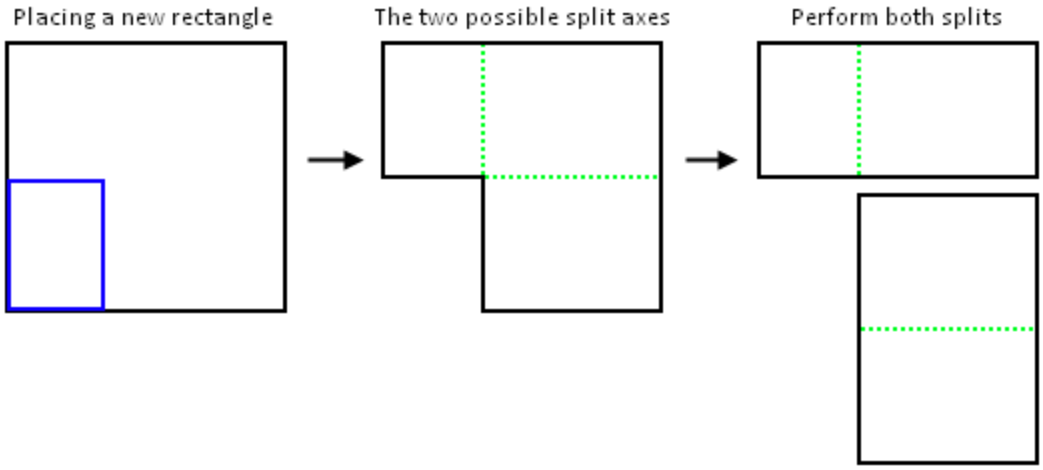
„Maximální“ v názvu odkazuje na vlastnost, že tyto nové dva obdélníky F_1 a F_2 jsou vytvořeny tak, aby byly jejich strany nejdelší možné v každém směru. To znamená, že se na každé straně dotýkají buď okraje kontejneru, nebo nějakého předmětu již uloženého v kontejneru.

Provedení tohoto dělení dává speciální vlastnost množině F. Množina F je množinou maximálních volných obdélníků reprezentující volné místo v kontejneru při určitém kroku algoritmu maximálních obdélníků. Potom pro libovolný předmět R, který je podmnožinou množiny sjednocení všech volných obdélníků, existuje takový volný obdélník F_i, pro který platí, že předmět R je jeho podmnožinou a může do něj tedy být vložen.

Díky této vlastnosti lze zaručit, že pokud je uvažována potencionální pozice pro uložení předmětu, jsou uvažovány všechny obdélníky volného místa F_i v pořadí, pokud tedy lze předmět vložit na určité místo v kontejneru, nemůže se stát, že bychom toto místo vynechali a předmět neuložili.

Odpadá vlastnost, že obdélníky volného místa F_i se po rozdělení nepřekrývají. Tato vlastnost způsobovala problémy při ukládání předmětu do kontejneru. Toto je způsobeno tím, že po vložení předmětu R do obdélníku volného místa F_i je nutné zkontrolovat a aktualizovat všechny ostatní obdélníky volného místa F_j z množiny F, pro které platí, že průnik ukládaného předmětu R a těchto obdélníků F_j není nula, tedy že do nich zasahuje, jinak se datová struktura dostane do rozporu.

Toto se jednoduše provede tak, že se projdou všechny obdélníky volného místa F_j a protnou se předmětem R, čímž vznikne množina nových obdélníků volného místa. Po tomto kroku se v množině F mohou nacházet zdegenerované a/nebo nemaximální obdélníky. Proto se projde, každý obdélník volného místa F_i z množiny F a je odstraněn, pokud se najde jiný obdélník F_j z množiny F, platí $i \neq j$, jehož je obdélník F_i podmnožinou.

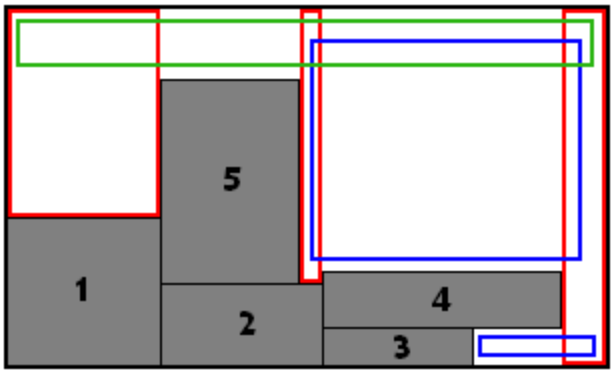


Obr. 2-4 Rozdělení volného místa pro algoritmus maximálních obdélníků. Oba obdélníky volného místa vpravo jsou uloženy do množiny F. (4)

2.4.1 Maximal Rectangles Bottom-Left (MAXRECTS-BL)

Algoritmus MAXRECTS-BL je odlišný od algoritmů popsaných výše. Říká se mu také Tetris algoritmus. Heuristické pravidlo tohoto algoritmu je jednoduché. Orientovat a umístnit každý předmět tak, aby byla vzdálenost ve směru y jeho horní strany co možná nejmenší. Pokud je zde více možných pozic pro uložení předmětu, volí se taková, kde je vzdálenost ve směru x co možná nejmenší. Při aplikaci toho pravidla na algoritmus maximálních obdélníků je výsledkem právě algoritmus MAXRECTS-BL.

Na obrázku 2-5 je zobrazen jednoduchý výstup MAXRECTS-BL algoritmu. Maximální obdélníky volného místa jsou zobrazeny červeně, zeleně a modře a všechny jsou v množině F. Pro jasnost jsou částečně zmenšeny.



Obr. 2-5 Příklad skládání předmětů pomocí MAXRECTS-BL algoritmu. (4)

2.4.2 Maximal Rectangles Best Area Fit (MAXRECTS-BAF)

Při výběru obdélníku volného místa v datové struktuře maximálních obdélníků může být použito stejné heuristické pravidlo jako u gilotinových algoritmů. U algoritmu MAXRECTS-

BAF je vybrán obdélník volného místa F_i z množiny F takový, který je nejmenší v kontejneru, do nějž lez uložit následující předmět R.

Pokud se najde více takových shodných míst, použijeme pravidlo Best Short Side Fit.

2.4.3 Maximal Rectangles Best Short Side Fit (MAXRECTS-BSSF)

Při uvažování rozdílů délek stran mezi předmětem R a obdélníkem volného místa F_i pracuje algoritmus MAXRECTS-BSSF shodně jako gilotinový algoritmus GUILLOTINE-BSSF. Algoritmus zvolí uložení předmětu R do takového obdélníku volného místa F_i, aby minimum rozdílů délek w_f-w a h_f-h bylo co možná nejmenší. Jinými slovy minimalizuje velikost menšího rozdílu délek stran.

2.4.4 Maximal Rectangles Best Long Side Fit (MAXRECTS-BLSF)

Pravidlo algoritmu MAXRECTS-BLSF je naprosto analogické. Algoritmus zvolí uložení předmětu R do takového obdélníku volného místa F_i, aby maximum rozdílů délek w_f-w a h_f-f bylo co možná nejmenší. Jinými slovy minimalizuje velikost většího rozdílu délek stran.

2.4.5 Účinnost MAXRECTS algoritmů

Analyzování účinnosti algoritmů, které jsou založeny na datové struktuře MAXRECTS není snadné a přímočaré.

Po uložení každého předmětu a po jeho překřížení prvky množiny F a po vytvoření množiny nových potenciálně maximálních obdélníků, projdeme přes každý pár prvků množiny F k omezení přebytečných obdélníků volného místa. Tento krok algoritmu je časově nejnáročněji.

Na základě tohoto poznatku je velmi důležité znát tempo růstu množiny F, aby se dala odhadnout aktuální složitost těchto algoritmů.

2.4.6 Maximal Rectangles Contact Point (MAXRECTS-CP)

Tento algoritmus je oproti těm již zmíněným unikátní.

U tohoto algoritmu je snaha najít místo pro uložení předmětu R do takové pozice, kde je délka obvodu předmětu, na které se předmět dotýká okraje kontejneru nebo dříve uložených předmětů, maximální. V tomto algoritmu je uvažováno pouze stabilní ukládání předmětů do levého spodního rohu. Jinak se tomuto algoritmu říká algoritmus dotýkajícího se obvodu.

Jediným problémem oproti ostatním MAXRECTS algoritmům je ten, že pro dosažení této metody je potřeba projít množinou všech již uložených předmětů. Je to lineární krok, který je nutno provést pro každý ukládaný předmět.

Nicméně jelikož omezování přebytečných obdélníků volného místa je časově mnohem náročnější, čas potřebný na tento krok je zanedbatelný. Spotřeba místa pro tento algoritmus je naprosto stejná jako u ostatních MAXRECTS algoritmů.

2.5 Panoramatické algoritmy

Protože algoritmy maximálních obdélníků zahrnují zdlouhavou manipulaci k údržbě množiny maximální obdélníků volného místa, byla navržena zjednodušená datová struktura, která může být také použita pro realizaci heuristiky levého dolního rohu. (7)

Datová struktura u panoramatických algoritmů je ztrátová, stejně jako u policových algoritmů, protože neumí perfektně sledovat volná místa v kontejneru a může označit nějaká nezaplněná

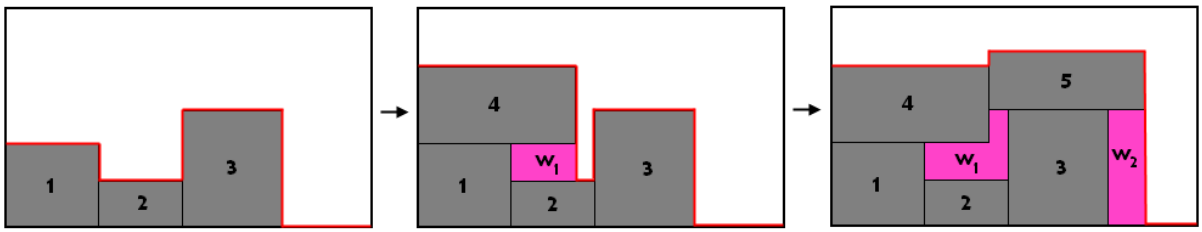
místa v kontejneru jako zaplněná. Na druhou stranu panoramatický algoritmus produkuje výsledky uložení předmětů mnohem rychleji než algoritmy používající datovou strukturu maximálních obdélníků.

Způsob, kterým panoramatické algoritmy pracují, je ten, že udržují pouze množinu horizontálních (panoramatických) hran, tvořených pouze nejvyššími hranami již uložených předmětů. Řídit tuto množinu je velmi jednoduché. Tato množina roste lineárně s počtem již uložených předmětů.

2.5.1 Skyline Bottom-Left (SKYLINE-BL)

Panoramatická datová struktura dovoluje uskutečnit stejnou heuristiku levého dolního rohu jako u MAXRECTS-BL algoritmu. Jediným rozdílem je, že se část skladovací efektivity vymění za vyšší časovou efektivitu.

U tohoto algoritmu je předmět R skládán zarovnaný doleva a je na vrchu takové panoramatické úrovně, na které je horní strana předmětu co nejníže. Protože je možno předmět otočit, nemusí se jeho horní strana nacházet v absolutně nejnižší pozici.



Obr. 2-6 Jednoduché skládání předmětů pomocí algoritmu SKYLINE-BL. (4)

2.5.2 Skyline Best Fit (SKYLINE-BF)

Panoramatická datová struktura je náchylná ke ztrátě informací o volném místě v kontejneru, je snaha tomu zabránit. Toho lze dosáhnout pomocí SKYLINE-BF varianty. U této varianty je pro každé volné místo, kam může být předmět uložen, spočtena celková oblast kontejneru, kterou ztratíme, pokud na toto místo předmět uložíme. Následně je vybrána oblast s nejmenší ztrátou. Pokud nastane shoda, řídíme se pravidlem levého dolního rohu.

2.5.3 The Waste Map Improvement (-WM)

Protože je snadné spočítat obdélníky volného místa, které ztratíme při uložení předmětu na vršek předmětu uloženého dříve, můžeme použít gilotinovou datovou strukturu k zachování tohoto volného místa a můžeme ji použít jako sekundární datovou strukturu.

3 Obecné zlepšující metody

V této kapitole budou zvážený metody, které zvyšují výkon skládání předmětů bez ohledu na to, jaký algoritmus je použit pro samotné skládání. U všech výše zmíněný algoritmů předpokládáme online BPP. Skládají všechny předměty v pořadí, ve kterém přichází na vstup, a nikdy nehýbají s předměty, které byly již uloženy. Nepřipouští se žádné hledání podílející se na vytváření možností. Tyto druhy omezení značně zjednodušují komplexnost algoritmů a snižují úsilí potřebné k jejich realizaci. Nevýhodou je, že kvalita provedeného ukládání předmětů bude v nejhorším případě hodně nízká. V této kapitole budou zváženy metody, které mohou být použity pro značné zlepšení situace.

3.1 Výběr cíleného kontejneru

Ještě nebylo řečeno, jak algoritmy fungují, pokud se nevejdou všechny předměty do jednoho kontejneru a musí se tedy použít více kontejnerů. Tato pravidla jsou velmi podobná heuristice, která byla použita při výběru police u policových algoritmů.

U Bin Next Fit (-BNF) algoritmu je otevřený pouze jeden kontejner, do kterého skládáme předměty. Pokud následující předmět na vstupu již nelze vložit do tohoto kontejneru, je tento kontejner uzavřen a dále nebude brán v potaz. Otevře se nový kontejner.

Všechny algoritmy, které byly zmíněny, jsou algoritmy typu –BNF, protože pracují pouze s jedním kontejnerem.

U Bin First Fit (-BFF) algoritmu jsou kontejnery uvažovány v takovém pořadí, v jakém byly otevřeny. Předmět je uložen do kontejneru s nejnižším číslem indexu, do kterého ho lze uložit.

U Bin Best Fit (-BBF) algoritmu je předmět uložen do takového kontejneru, který nejlépe splňuje kritéria, podle kterých se algoritmus rozhoduje mezi možnými místy pro uložení předmětu.

Analogicky jako u policových algoritmů je stanoven Bin Worst Fit algoritmus. Který však není v tomto případě příliš optimální.

3.2 Setřídění vstupu

Jednoduchou metodou pro zvýšení výkonu online BPP algoritmů je jednoduché setřídění pořadí podle nějakého kritéria před samotným procesem ukládání předmětů. Protože je tento krok před samotným procesem skládání, nevyžaduje žádné změny běžného postupu tohoto procesu, což jej činí velmi praktickým. Samozřejmě toho můžeme dosáhnout pouze, pokud dopředu známe celou množinu předmětů, které budou ukládány.

Existuje několik rozdílných metod, které lze použít jako srovnávací funkci pro běžný třídící postup. Jsou dány dva předměty $R_a=(w_a, h_a)$ a $R_b=(w_b,h_b)$, pro které platí $w_a \leq h_a$ a $w_b \leq h_b$. Tyto předměty mohou být porovnány následovně.

Setřídění podle plochy. Předmět R_a je zařazen před předmět R_b, pokud platí $w_a h_a < w_b h_b$. Tato metoda se nazývá –ASCA. Při obrácení podmínky se metoda nazývá –DESCA.

Setřídění podle délky kratších stran předmětů, následované srovnáním delších stran předmětů. Předmět R_a je zařazen před předmět R_b, pokud platí $w_a < w_b$ nebo pokud je $w_a = w_b$ a zároveň $h_a < h_b$. Tyto metody se nazývají -ASCSS a –DESCSS (při obrácení podmínky).

Setřídění podle délky delších stran předmětů, následované srovnání kratších stran předmětů. Předmět Ra je zařazen před předmět Rb, pokud platí $h_a < h_b$ nebo pokud je $h_a = h_b$ a zároveň $w_a < w_b$. Tyto metody se nazývají -ASCLS a –DESCLS (při obrácení podmínky).

Setřídění podle obvodu. Předmět R_a je zařazen před předmět R_b, pokud platí $w_a+h_a < w_b+h_b$. Tyto metody se nazývají -ASCPERIM a –DESCPERIM (při obrácení podmínky).

Setřídění podle rozdílu šířky a výšky předmětu. Předmět R_a je zařazen před předmět R_b, pokud platí $|w_a-h_a| < |w_b-h_b|$. Tyto metody se nazývají -ASCDIFF a –DESCDIFF (při obrácení podmínky).

Setřídění podle poměru šířky a výšky předmětu. Předmět R_a je zařazen před předmět R_b, pokud platí $w_a/h_a < w_b/h_b$. Tyto metody se nazývají -ASCRATIO a –DESCRATIO (při obrácení podmínky).

3.3 Nejlepší globální možnost

Většina uvažovaných metod, má strukturu, která může být představena následovně.

Pro následující předmět na vstupu R‘ z množiny předmětů na vstupu R, je dána množina možností uložení předmětu S. Dále je dána vyhodnocovací funkce C: S x R definovaná heuristickým pravidlem dané metody. Pomocí této vyhodnocovací funkce je nalezena nejlepší možnost S‘, podle které bude předmět R‘ uložen. Množina možností S může být dána následovně.

U policových algoritmů je množina S množinou všech polic, do kterých lze předmět R‘ uložit.

U gilotinových algoritmů je množina S množinou všech obdélníků volného místa F násobenou dvěma. To odkazuje na dvě platné možnosti položení předmětu R‘.

U algoritmů maximálních obdélníků je množina S množinou všech maximálních obdélníků násobenou dvěma. To odkazuje na dvě platné možnosti položení předmětu R‘.

U panoramatických algoritmů je množina S množinou panoramatických úrovní násobenou dvěma. To odkazuje na dvě platné možnosti položení předmětu R‘.

Existuje přirozené rozšíření tohoto optimalizačního pravidla, díky kterému lze dosáhnout lepšího uložení předmětů. Při každém kroku ukládání předmětů jde algoritmus přes každý předmět na vstupu a přes všechna možná uložení pro tento předmět. Následně je vhodnost každého uložení bodově ohodnocena podle vybraného heuristického pravidla pomocí vyhodnocovací funkce. Poté je vybrána možnost s největším bodovým ziskem. Jelikož místo následujícího předmětu vybíráme z množiny ten globálně nejlepší ze zbývajících předmětů, je tomuto pravidlu dána přípona –GLOBAL. U tohoto rozšíření nezáleží na seřazení předmětů na vstupu, vždy se vybere ten nejlepší bez ohledu na jeho index, proto není toto seřazení provedeno.

4 Policové algoritmy a jejich implementace do jazyku pascal

Tato kapitola je blíže zaměřena na policové algoritmy. Obsahuje popis proměnných, funkcí a procedur používaných v policových algoritmech. Dále obsahuje vývojové diagramy jednotlivých policových algoritmů. Jednotlivé policové algoritmy byly následně implementovány do programovacího jazyku pascal. Po naprogramování algoritmů bylo provedeno měření ukládání předmětů.

Práce je zaměřena na policové algoritmy z důvodu jejich širšího využití. Na rozdíl od ostatních skupin algoritmů lze tuto skupinu algoritmů použít i vertikálně. Tedy ne pouze pro ukládání předmětů na dno kontejneru, na paletu nebo pro rozložení předmětů na tabuli plechu (pro řezání), jinak řečeno „na zem“, ale také například pro ukládání předmětů do regálů. Regálové skladování patří celosvětově mezi nejpoužívanější typ skladování. Pro praktické využití policových algoritmů pro regálové skladování postačí myšlené čáry ohraničující jednotlivé police nahradit skutečnými regálovými policemi. Regály mají pevně danou šířku, lze však nastavit různé výšky jednotlivých polic přesně tak jako je tomu u policových algoritmů.

4.1 Funkce a procedury

V této podkapitole budou popsány procedury a funkce použité u policových algoritmů pro 2D obdélníkový Bin Packing Problem. U každé procedury nebo funkce bude krátký popis činnosti, seznam proměnných a vývojový diagram.

4.1.1 Funkce

U policových algoritmů je použita pouze jedna funkce.

Lze_ulozit

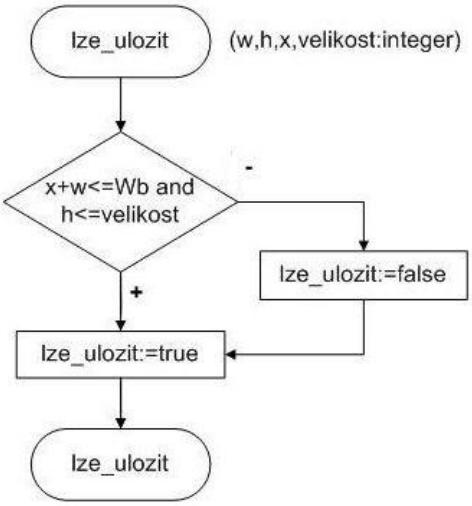
Funkce lze_ulozit je logickou funkcí testující, jestli je možno uložit předmět o daných rozměrech do dané police. Pokud ano, vrací hodnotu true, pokud ne, vrací hodnotu false.

Vstupní proměnné: w – šířka předmětu

h – výška předmětu

x – aktuální zaplnění dané police

velikost – výška (velikost) dané police



Obr. 4-1 Funkce lze_ulozit

4.1.2 Procedury

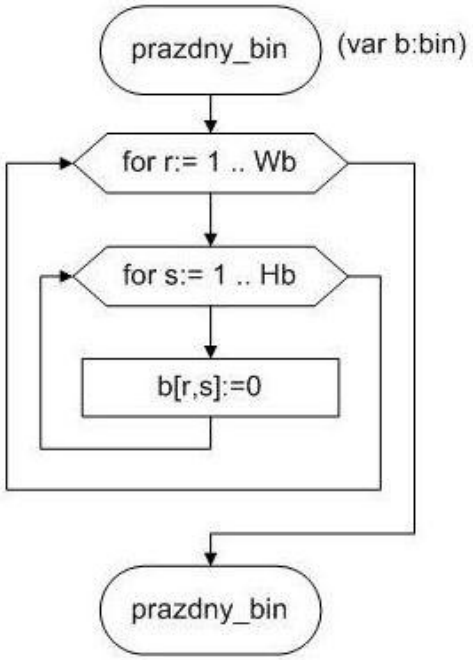
V policových algoritmech jsou vždy alespoň jednou použity následující procedury.

Prazdny_bin

Procedura prazdny_bin proběhne vždy na začátku každého algoritmu. Jedná se o nadefinování prázdného kontejneru pro ukládání předmětů. V našem případě se jedná o dvojrozměrné pole o rozměrech W_b a H_b. V tomto poli jsou všechny pozice nastaveny na nulu. Nula tedy značí prázdné neboli neobsazené místo.

Výstupní proměnné: b – kontejner (bin) = dvojrozměrné pole o rozměrech W_b a H_b

Lokální proměnné: r, s – pomocné proměnné pro for cyklus



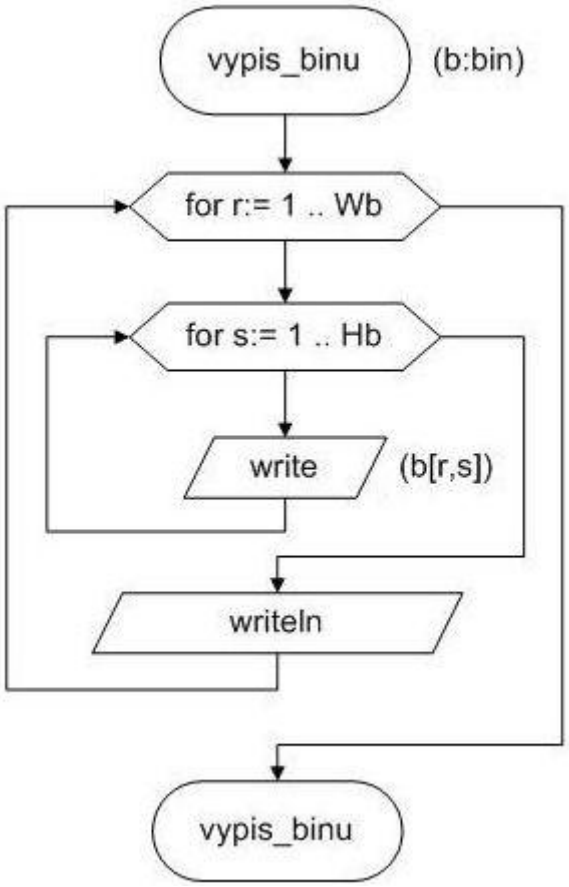
Obr. 4-2 Procedura prazdny_bin

Vypis_binu

Tato procedura zajišťuje výpis kontejneru na obrazovku. V průběhu ukládání předmětů lze tedy kdykoliv zobrazit aktuální zaplnění kontejneru.

Vstupní proměnné: b – kontejner (bin)

Lokální proměnné: r, s – pomocné proměnné pro for cyklus



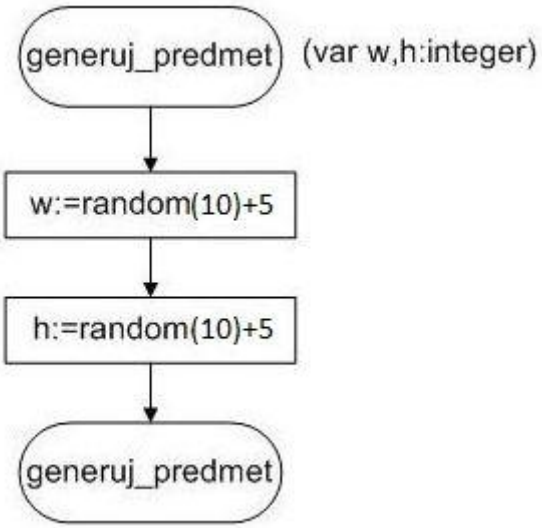
Obr. 4-3 Procedura vypis_binu

Generuj_predmet

Tato procedura generuje šířku w a výšku h předmětů, které se mají být uloženy do kontejneru.

Výstupní proměnné: w – šířka předmětu

h – výška předmětu



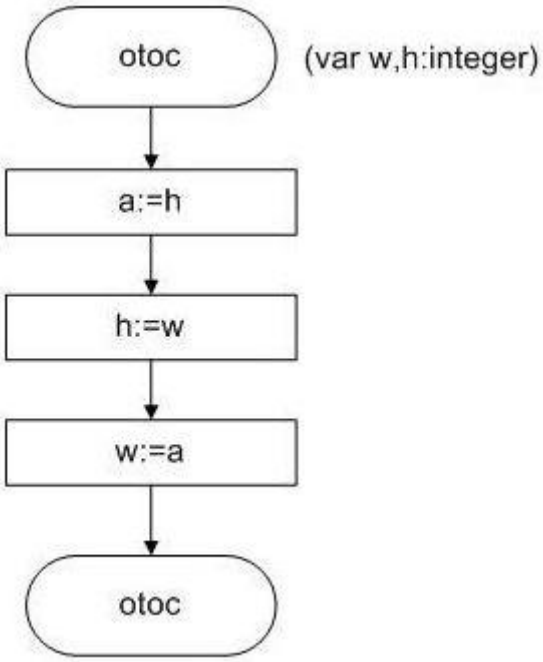
Obr. 4-4 Procedura generuj_predmet

Otoc

Tato procedura otáčí předmětem. To znamená, že prohazuje hodnoty šířky a výšky předmětu.

Výstupní proměnné: w – šířka předmětu
h – výška předmětu

Lokální proměnné: a – pomocná proměnná



Obr. 4-5 Procedura otoc

Uloz

Procedura uloz provádí vlastní uložení předmětu do kontejneru. V našem případě přepisuje na daných místech ve dvojrozměrném poli (kontejneru) hodnoty z nuly na číslo předmětu.

Vstupní proměnné: y – ypsilonová souřadnice dna dané police

w – šířka předmětu

h – výška předmětu

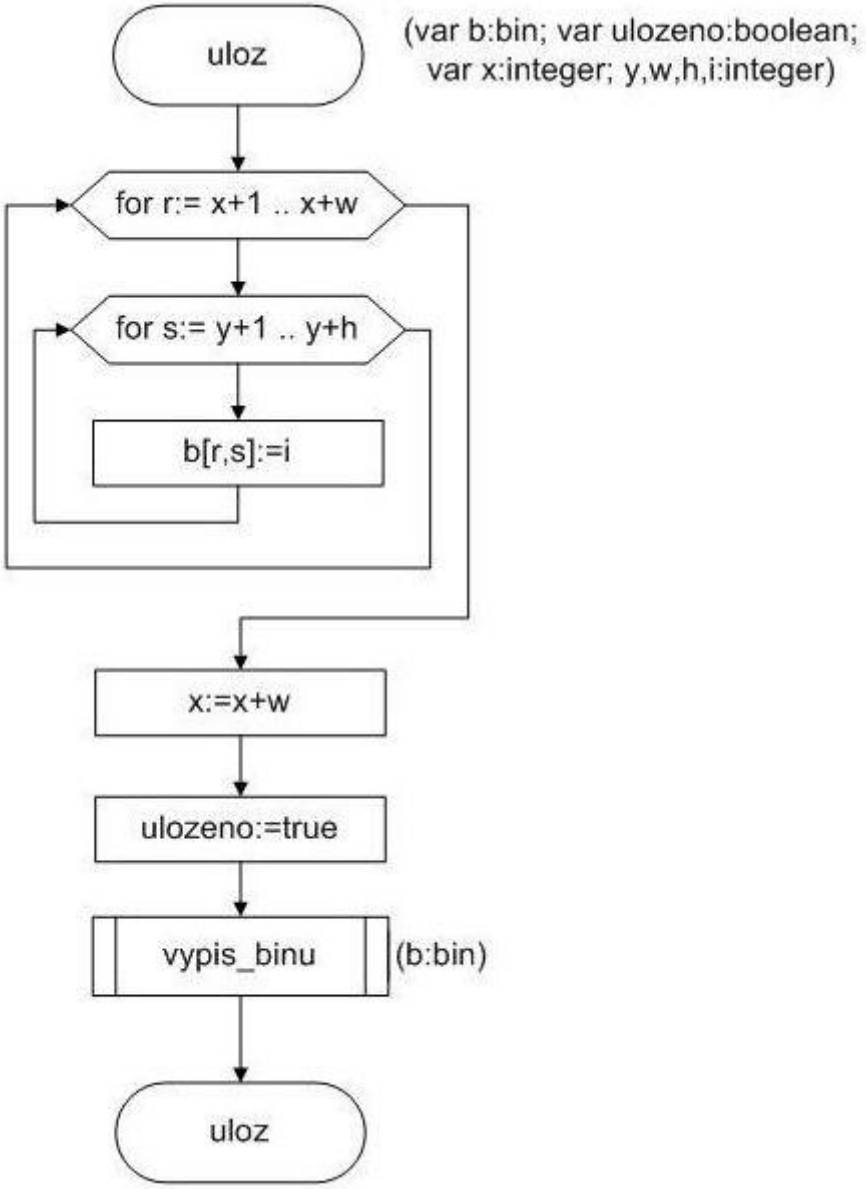
i – číslo předmětu

Lokální proměnné: r, s – pomocné proměnné pro for cyklus

Výstupní proměnné: x – aktuální zaplněnost dané police v ose x

b – kontejner (bin)

ulozeno – logická proměnná, při uložení předmětu má hodnotu true



Obr. 4.6 Procedura uloz

Vysledek

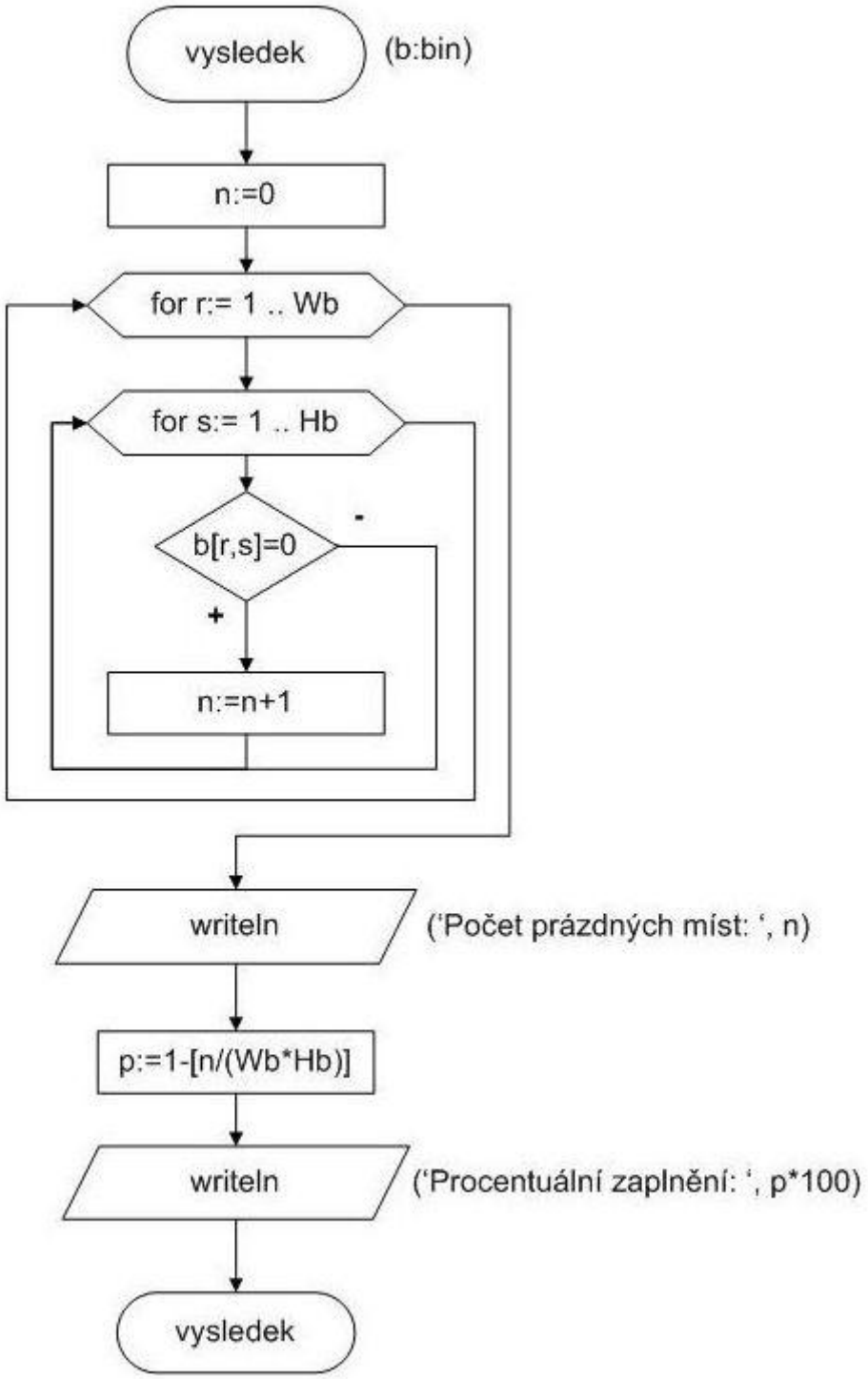
Procedura vysledek po uložení posledního předmětu spočítá všechna prázdná místa v kontejneru, tedy všechna místa s hodnotou nula. Následně pomocí této hodnoty a rozměrů kontejneru spočítá procentuální zaplnění kontejneru.

Vstupní proměnné: b – kontejner (bin)

Lokální proměnné: r, s – pomocné proměnné pro for cyklus

n – počet prázdných míst v kontejneru

p – procentuální zaplnění kontejneru



Obr. 4-7 Procedura vysledek

4.2 Policové algoritmy

V této kapitole budou popsány policové algoritmy. U každého algoritmu bude jeho popis se specifikacemi, výpis používaných proměnných a vývojový diagram.

4.2.1 Shelf Next Fit

Shelf Next Fit je nejjednodušší policový algoritmus. Tento algoritmus vždy pracuje pouze s poslední policí. Algoritmus otestuje, zdali je možné uložit předmět do dané police na výšku. Tím je docíleno minimalizace nevyužitého místa mezi horní hranou předmětu a stropem police. Pokud nelze předmět uložit na výšku je otočen a testuje se, zdali je možné uložit předmět na šířku. Pokud nelze předmět uložit ani na šířku je police uzavřena. Do uzavřených polic se tento algoritmus již nevrací. Následně algoritmus otestuje, jestli je v kontejneru dostatek místa pro vytvoření nové police. Pokud ano, je vytvořena nová police s předmětem uloženým na šířku, pokud ne, algoritmus končí.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

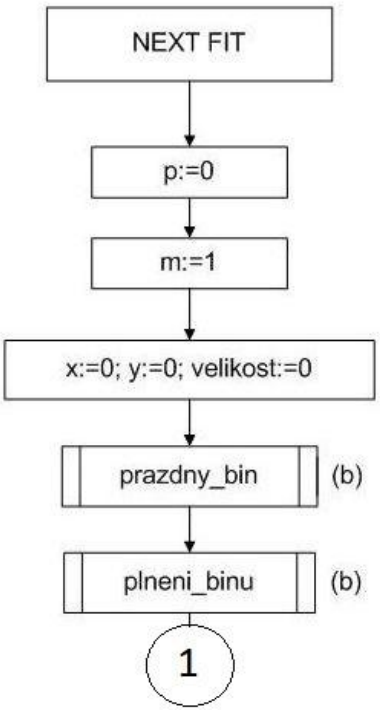
w – šířka předmětu

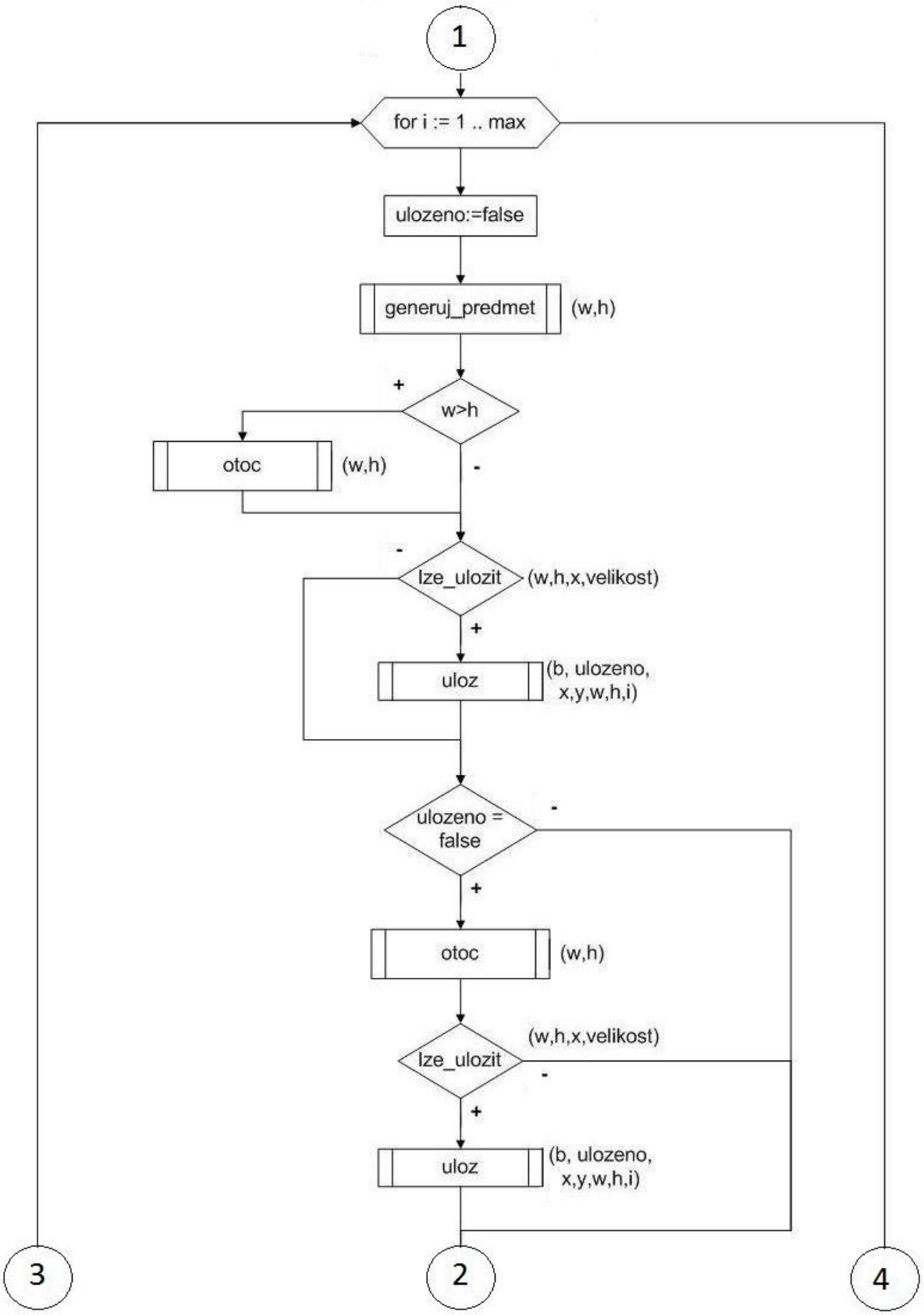
h – výška předmětu

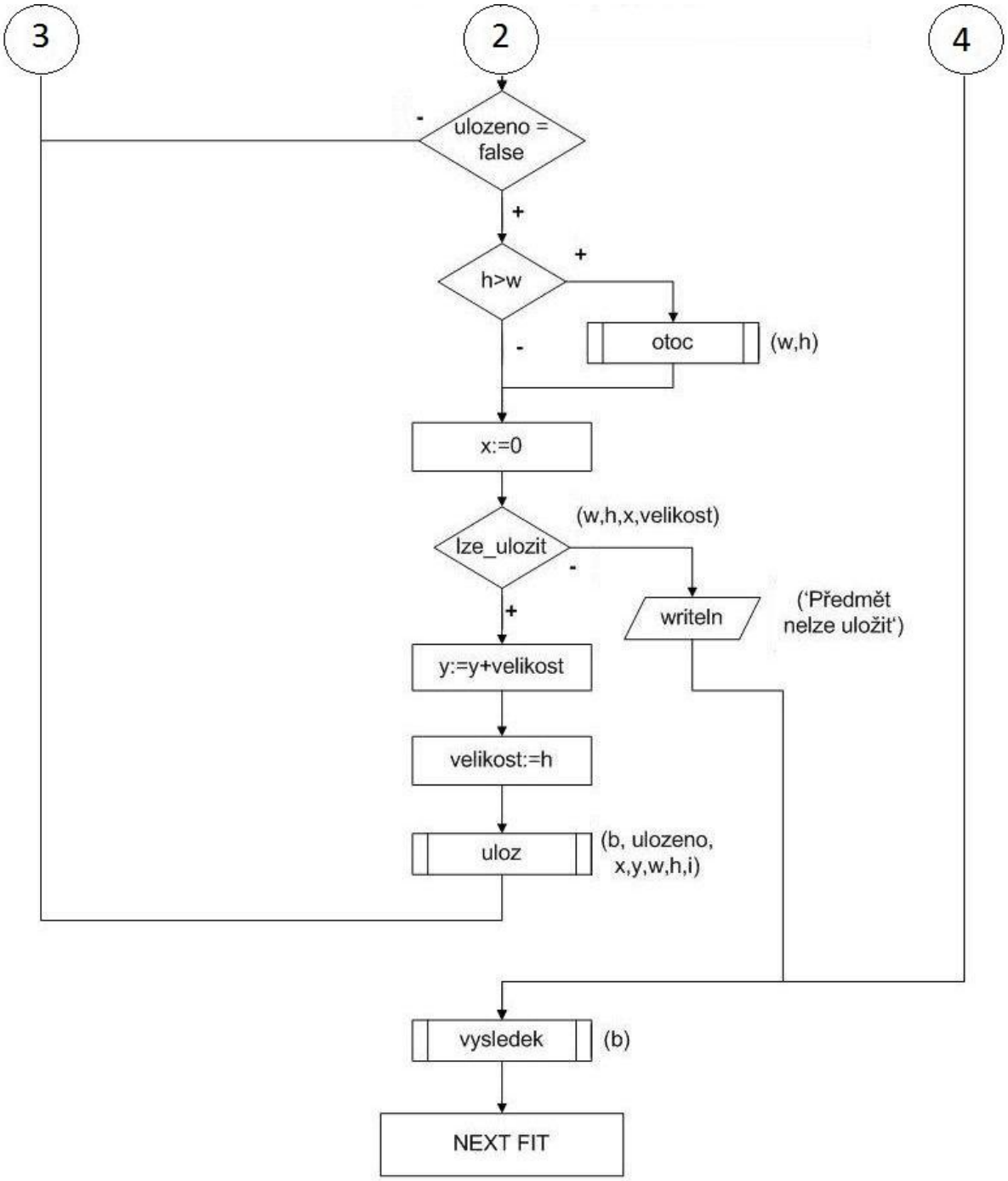
x – zaplněnost police (x-ová souřadnice)

y – ypsilonová souřadnice dna police

velikost – velikost (výška) police




1
for i := 1 .. max
ulozeno:=false
generuj_predmet
(w,h)
+
w>h
otoc
(w,h)
-
-
lze_ulozit
(w,h,x,velikost)
+
uloz
(b, ulozeno, x,y,w,h,i)
ulozeno = false
-
+
otoc
(w,h)
(w,h,x,velikost)
lze_ulozit
-
+
uloz
(b, ulozeno, x,y,w,h,i)
3
2
4



Obr. 4-8 Algoritmus Next Fit

4.2.2 Shelf First Fit

Největším nedostatkem Next Fit algoritmu je uzavírání polic pouze kvůli tomu, že do nich nelze uložit jeden předmět. Mnohem hospodárnější je udržovat si seznam uzavřených polic a jejich zaplněnost. Pokud se předmět uloží do jedné z uzavřených polic, ušetří se tím místo na polici otevřené. Pokud existuje více uzavřených polic, do kterých lze předmět uložit, zvolí se jedna z nich.

U First Fit algoritmu se prochází odspodu přes uzavřené police a hledá se nejspodnější police, do které lze předmět uložit. Algoritmus tedy postupuje od první police nahoru a testuje, je-li možnost předmět do dané police uložit na výšku, případně na šířku, pokud ano, uloží ho, pokud ne, postupuje do další police. Pokud nelze předmět uložit do žádné z již vytvořených polic, vytvoří se nová police, pokud to zbývající místo v kontejneru dovoluje.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

m – index police

p – počet polic

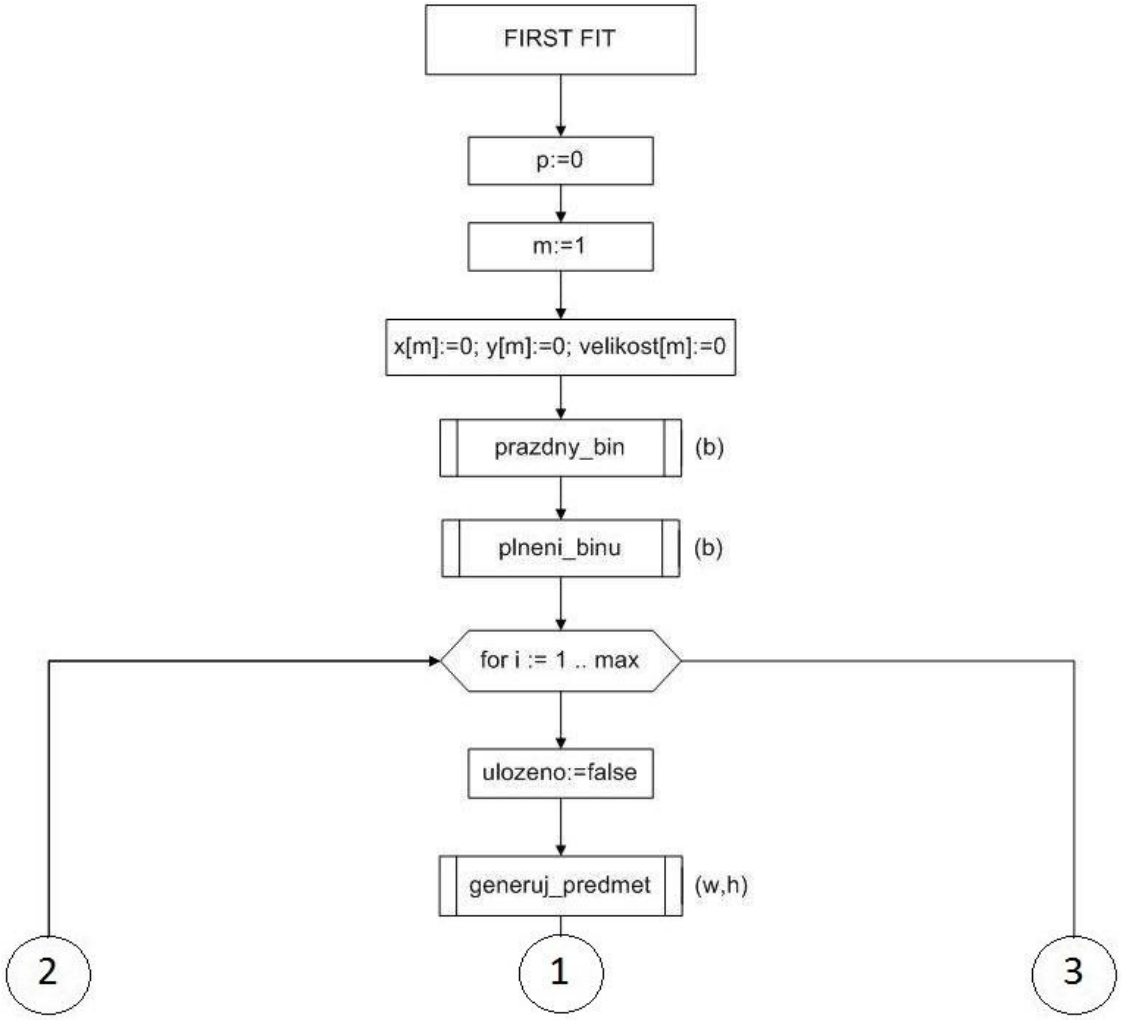
w – šířka předmětu

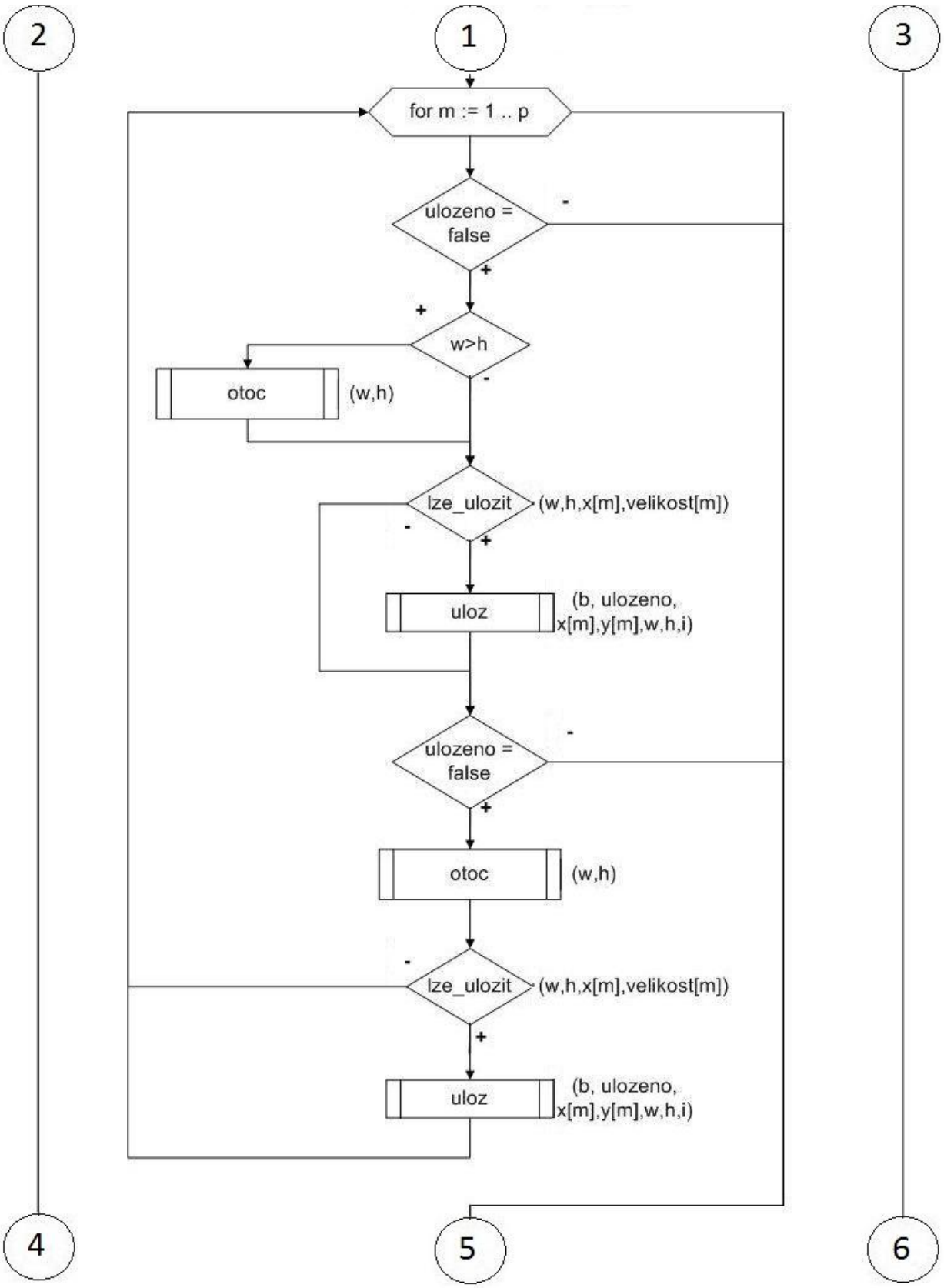
h – výška předmětu

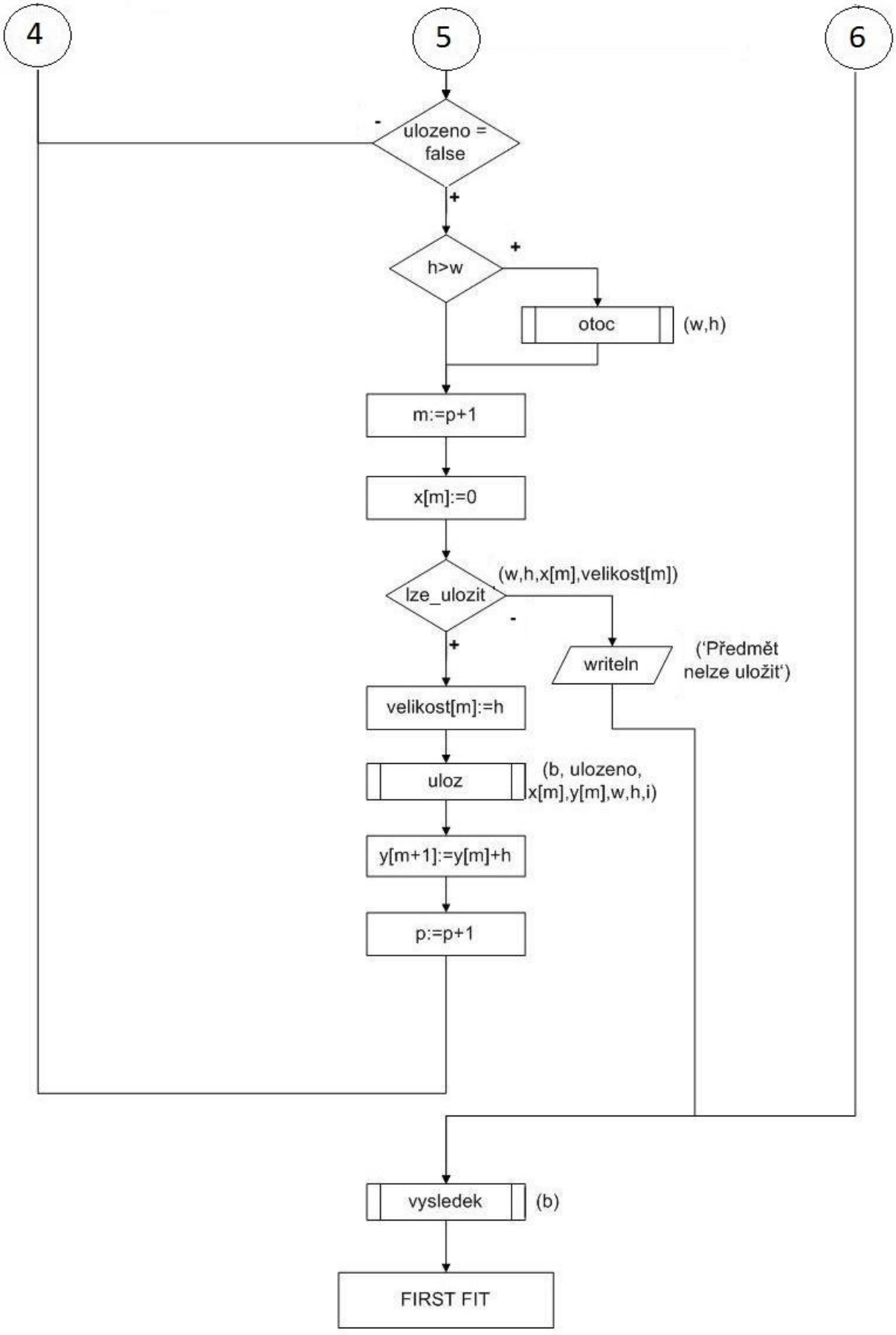
x – jednorozměrné pole zaplněnosti jednotlivých polic

y – jednorozměrné pole souřadnic spodků polic

velikost – jednorozměrné pole velikostí (výšek) polic




2
1
3
for m := 1 .. p
ulozeno = false
-
+
+
w>h
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
-
+
uloz
(b, ulozeno, x[m],y[m],w,h,i)
ulozeno = false
-
+
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
+
uloz
(b, ulozeno, x[m],y[m],w,h,i)
4
5
6



Obr. 4-9 Algoritmus First Fit

4.2.3 Shelf Best Height Fit

Best Height Fit algoritmus prochází všechny police jako First Fit algoritmus. Na rozdíl od First Fit algoritmu ale neuloží předmět na první volné místo do nejspodnější police. Ve všech policích, kam lze předmět uložit na výšku nebo na šířku, změří rozdíl mezi horní hranou předmětu a stropem police. Předmět bude uložen do police, kde je tento rozdíl nejmenší.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

navysku – logická proměnná, pokud je předmět otočen při ukládání na výšku, má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

m – index police

p – počet polic

w – šířka předmětu

h – výška předmětu

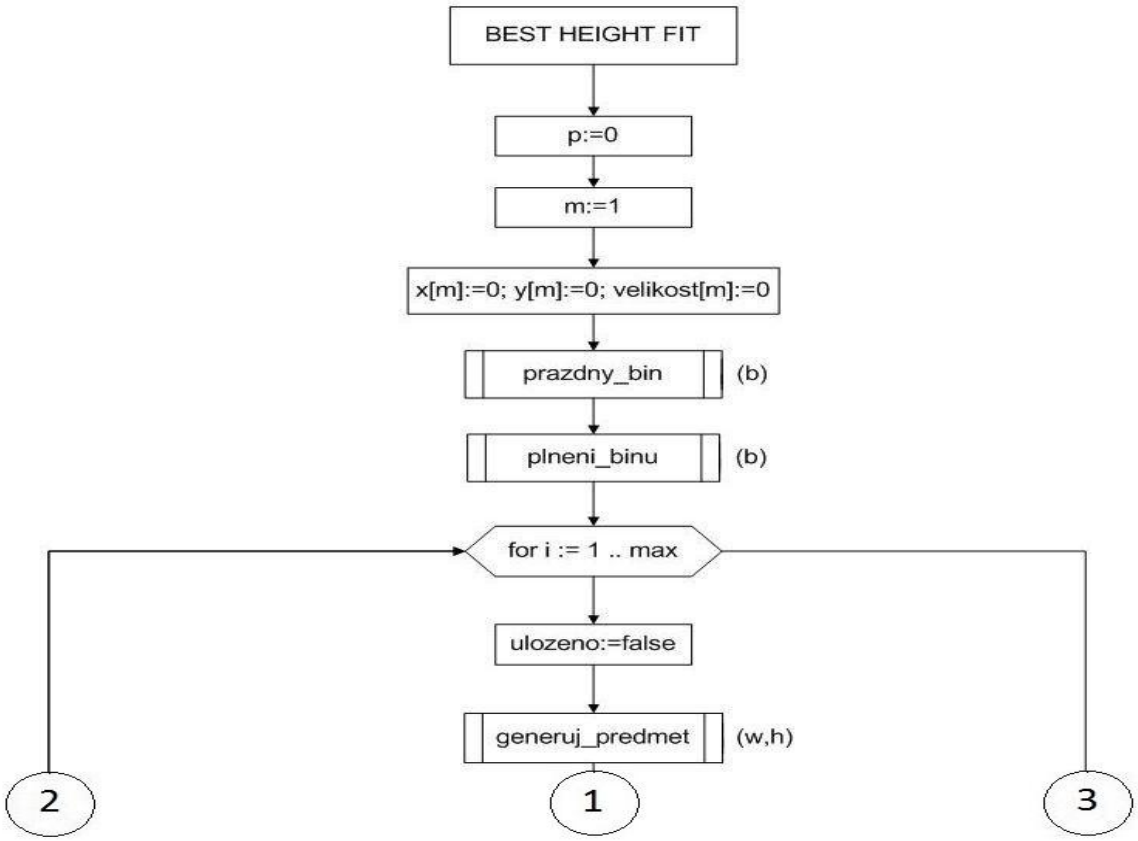
min_rozdil – nejmenší rozdíl mezi horní hranou předmětu a stropem police

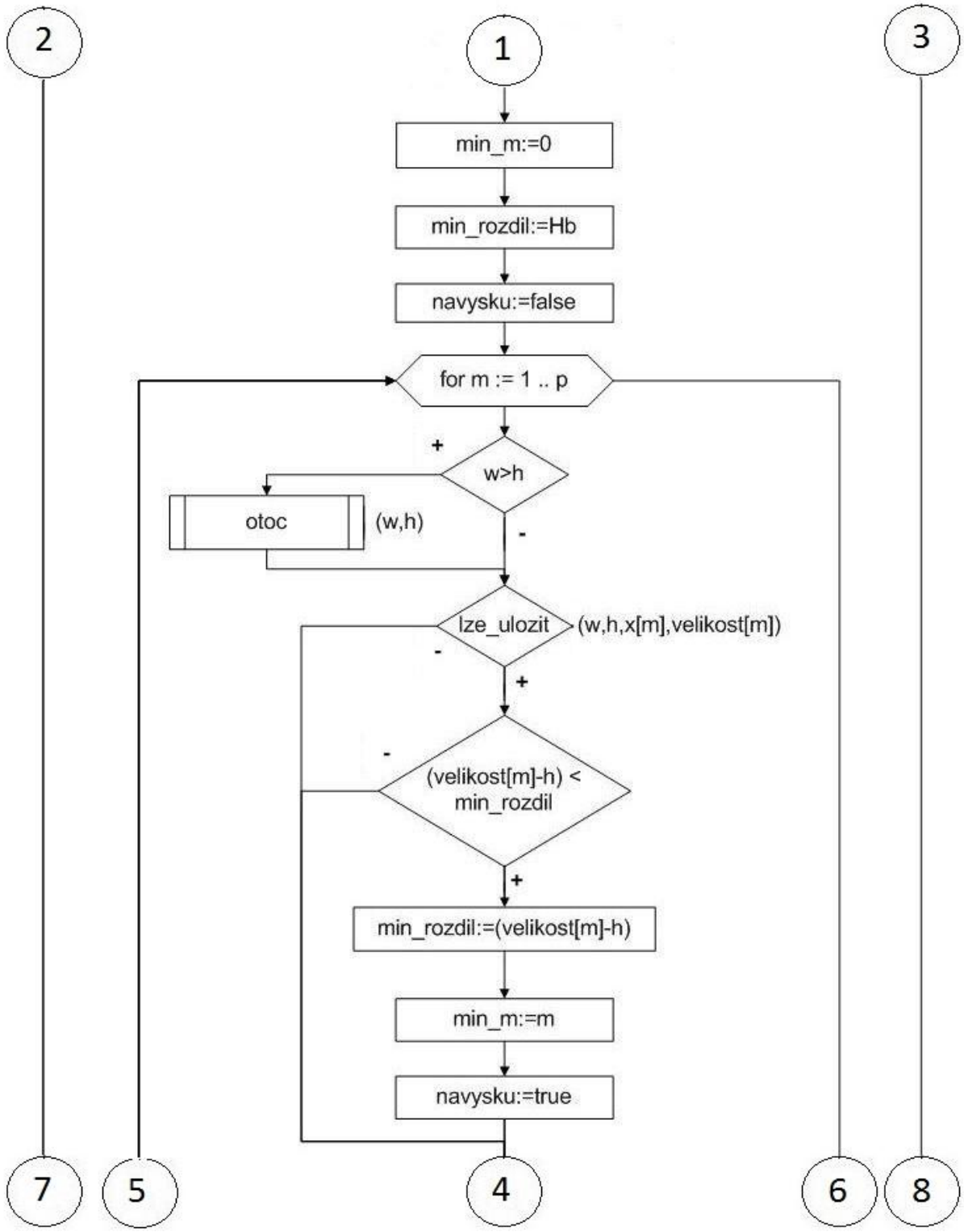
min_m – index police s nejmenším rozdílem

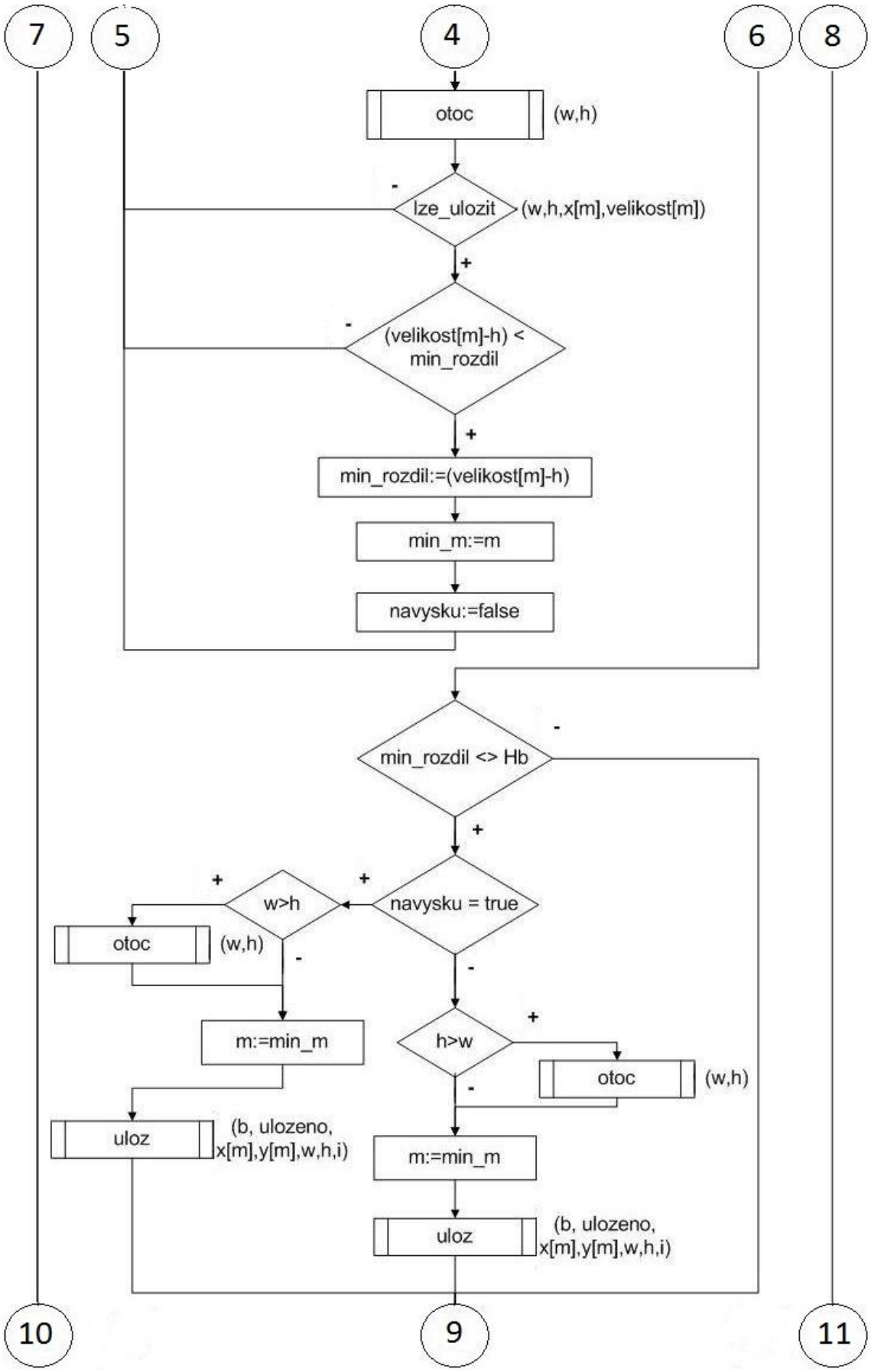
x – jednorozměrné pole zaplněnosti jednotlivých polic

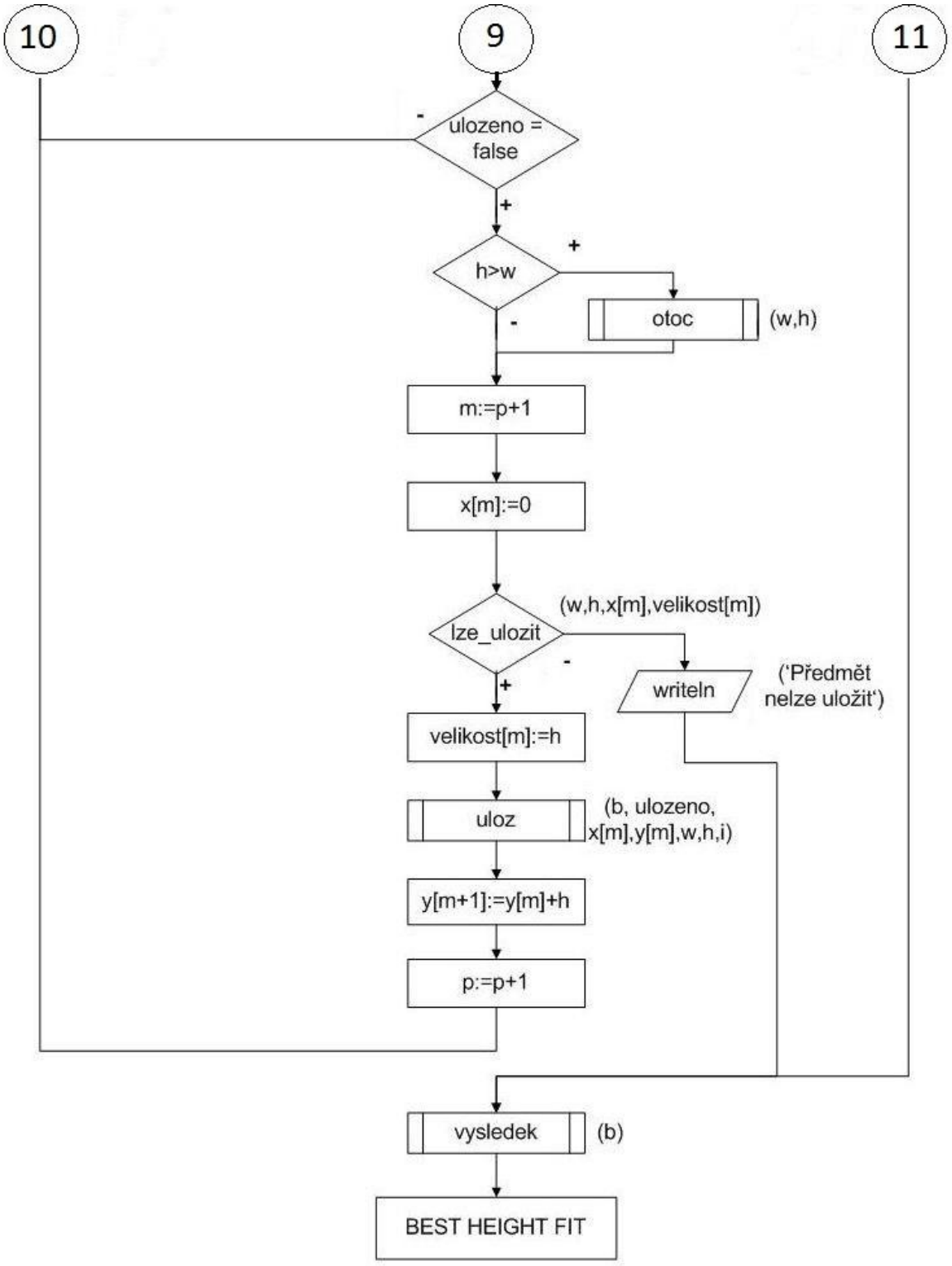
y – jednorozměrné pole souřadnic spodků polic

velikost – jednorozměrné pole velikostí (výšek) polic




2
1
3
min_m:=0
min_rozdil:=Hb
navysku:=false
for m := 1 .. p
+
w>h
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
-
+
-
(velikost[m]-h) <
min_rozdil
+
min_rozdil:=(velikost[m]-h)
min_m:=m
navysku:=true
7
5
4
6
8


7
5
4
6
8
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
(velikost[m]-h) <
min_rozdil
+
min_rozdil:=(velikost[m]-h)
min_m:=m
navysku:=false
min_rozdil <> Hb
-
+
+
+
w>h
navysku = true
otoc
(w,h)
-
-
m:=min_m
h>w
+
otoc
(w,h)
-
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
m:=min_m
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
10
9
11



Obr. 4-10 Algoritmus Best Height Fit

4.2.4 Shelf Best Widht Fit

Best Width Fit algoritmus prochází všechny police jako First Fit algoritmus. Na rozdíl od First Fit algoritmu ale neuloží předmět na první volné místo do nejspodnější police. Ve všech policích, kam lze předmět uložit na výšku nebo na šířku, změří zbývající volné místo v každé z polic. Předmět bude uložen do police, kde je toto zbývající místo nejmenší.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

navysku – logická proměnná, pokud je předmět otočen při ukládání na výšku, má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

m – index police

p – počet polic

w – šířka předmětu

h – výška předmětu

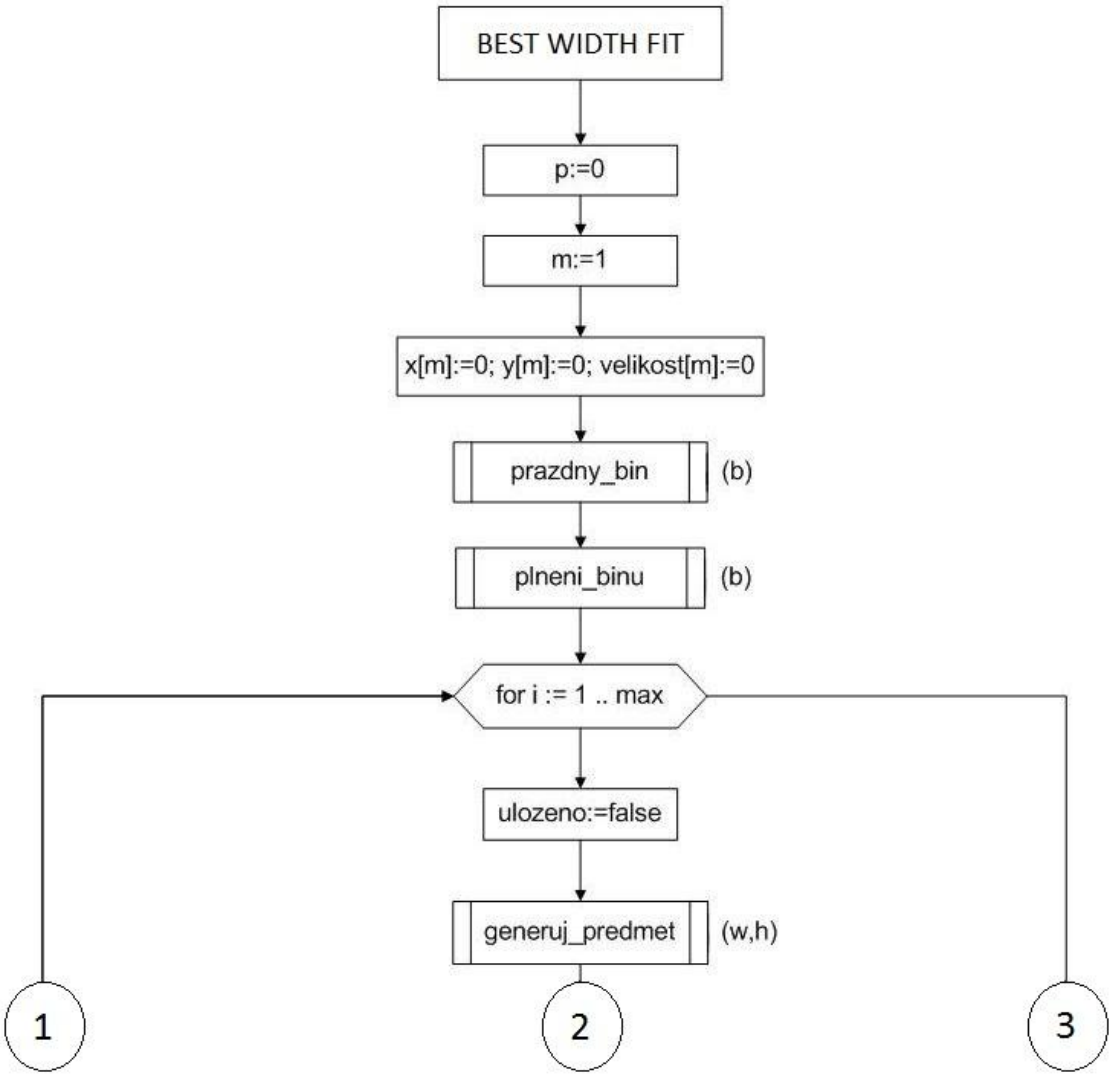
min_rozdil – nejmenší zbývající volné místo na polici

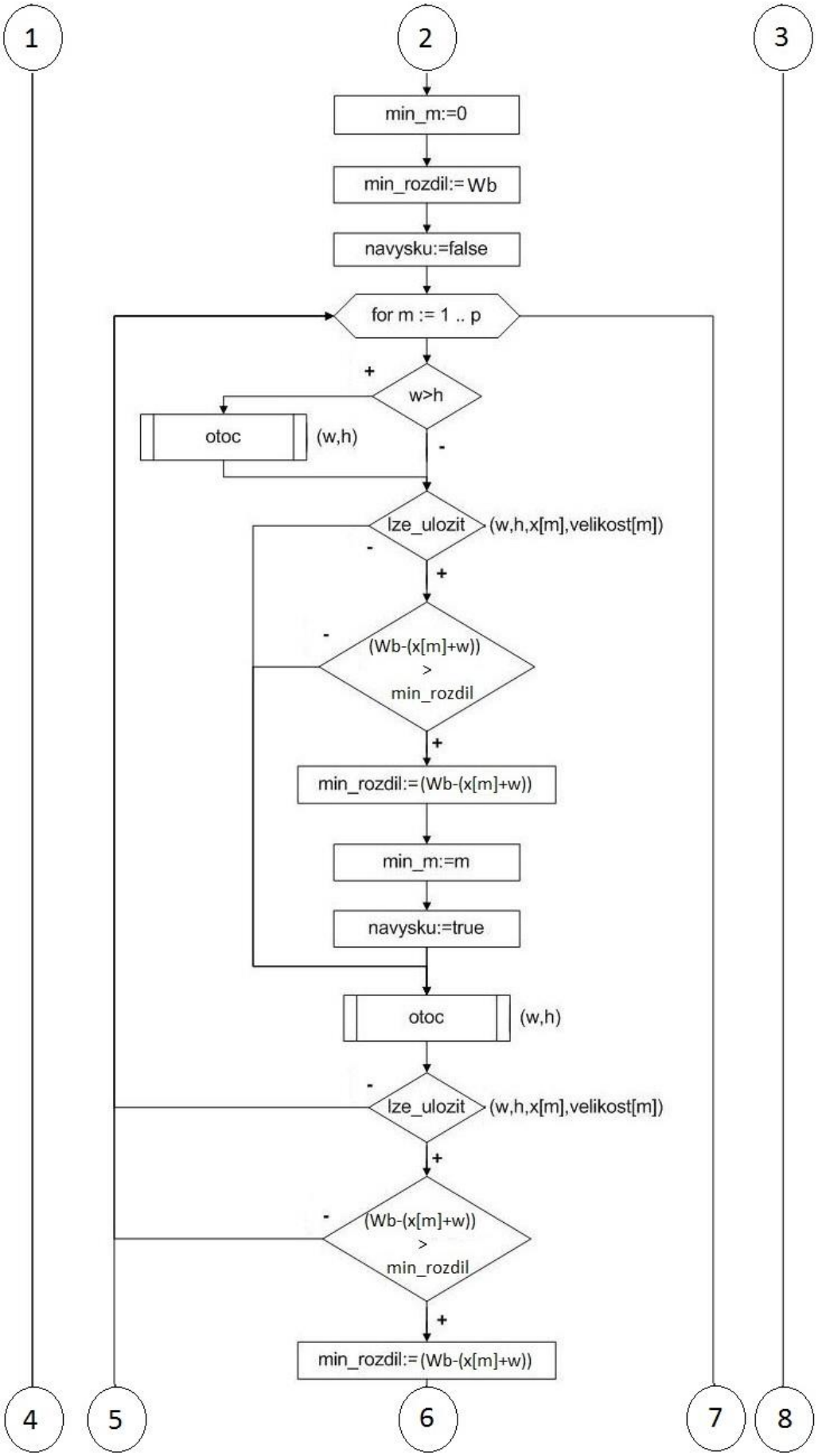
min_m – index police s nejmenším zbývajícím volným místem

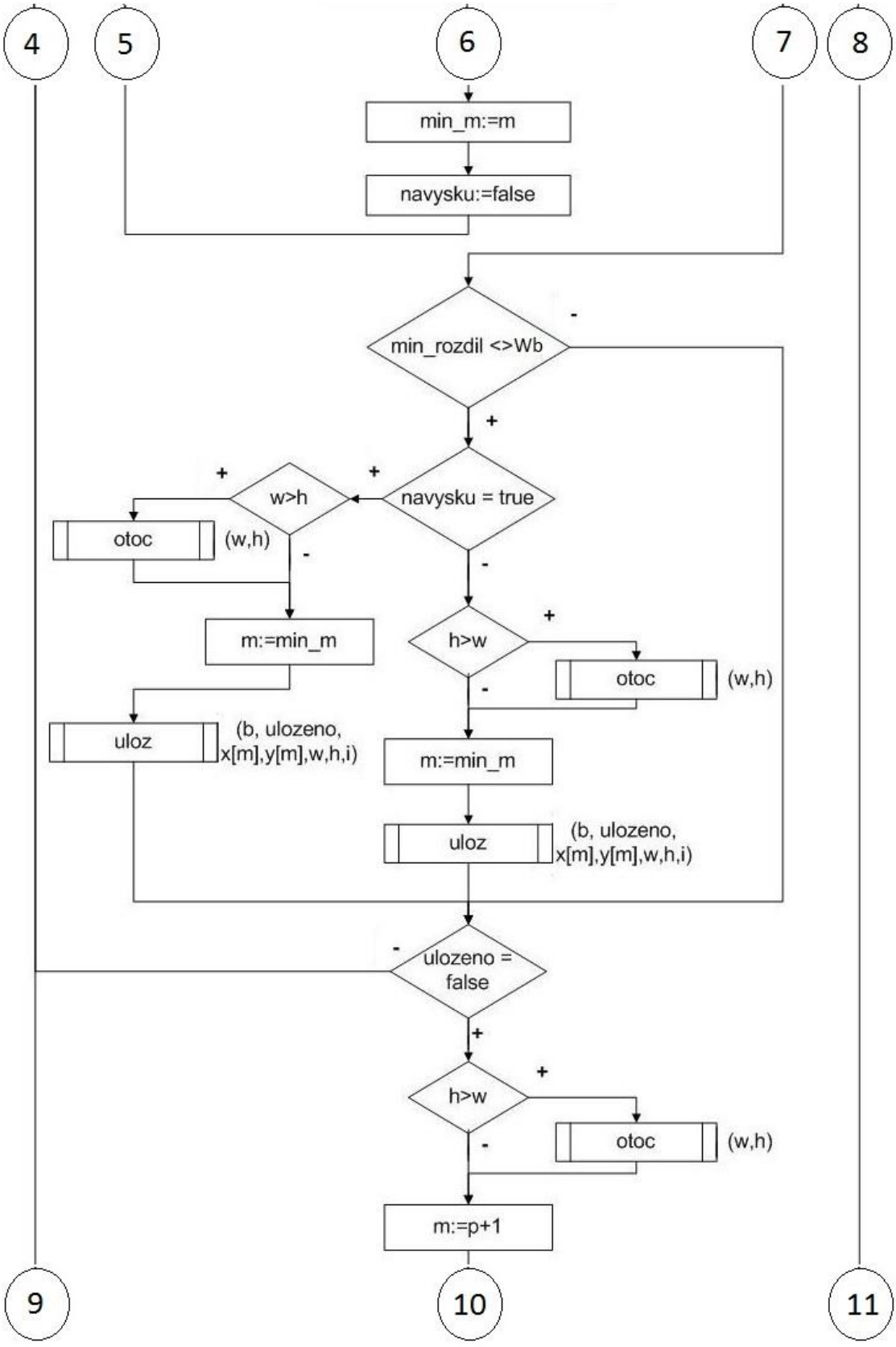
x – jednorozměrné pole zaplněnosti jednotlivých polic

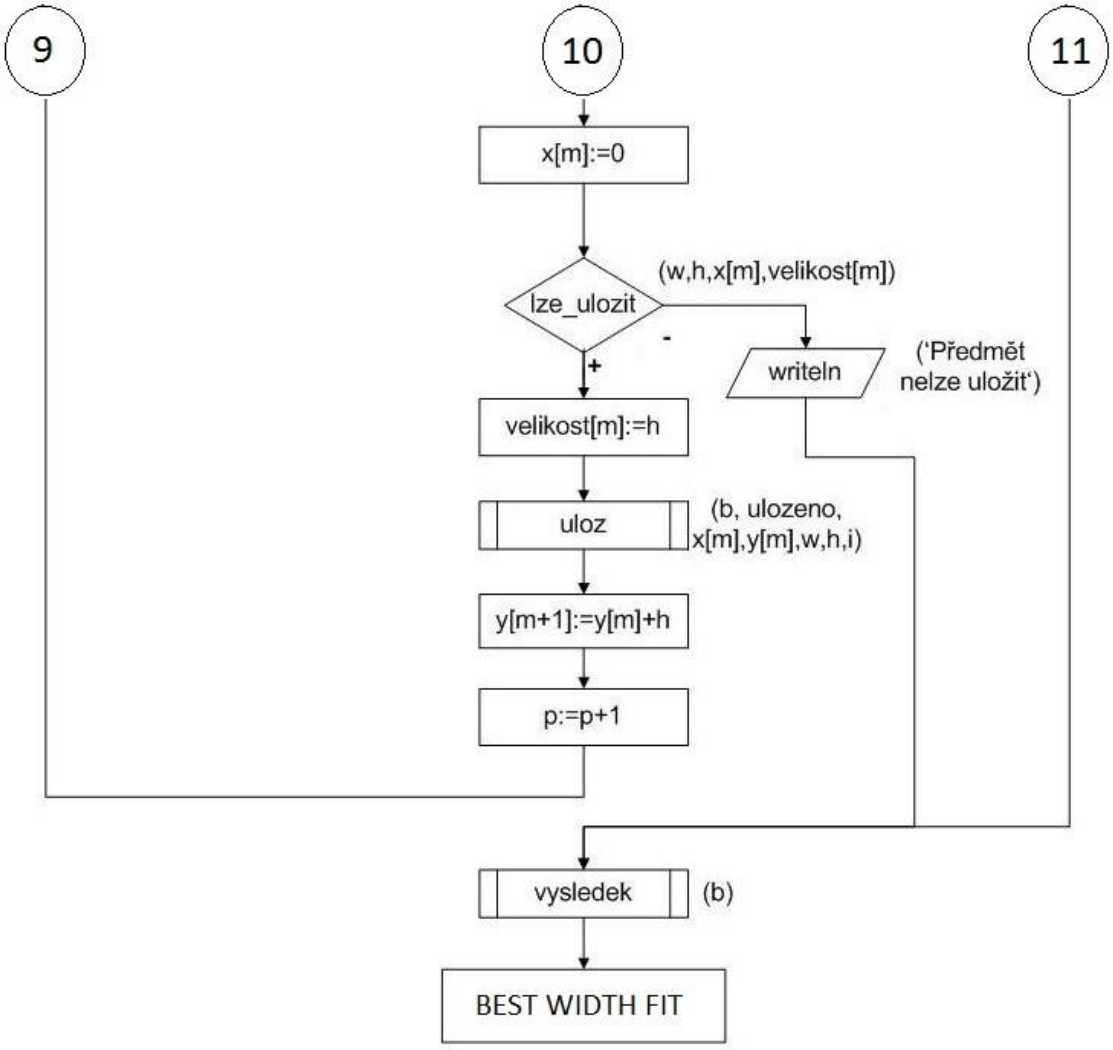
y – jednorozměrné pole souřadnic spodků polic

velikost – jednorozměrné pole velikostí (výšek) polic




1
2
3
min_m:=0
min_rozdil:= Wb
navysku:=false
for m := 1 .. p
+
w>h
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
-
+
-
(Wb-(x[m]+w))
>
min_rozdil
+
min_rozdil:=(Wb-(x[m]+w))
min_m:=m
navysku:=true
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
(Wb-(x[m]+w))
>
min_rozdil
+
min_rozdil:=(Wb-(x[m]+w))
4
5
6
7
8


4
5
6
7
8
min_m:=m
navysku:=false
min_rozdil <>Wb
-
+
navysku = true
+
w>h
+
otoc
(w,h)
-
-
m:=min_m
h>w
+
otoc
(w,h)
-
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
m:=min_m
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
-
ulozeno =
false
+
h>w
+
otoc
(w,h)
-
m:=p+1
9
10
11



Obr. 4-11 Algoritmus Best Width Fit

4.2.5 Shelf Best area fit

Algoritmus Best Area Fit vychází z předchozího algoritmu Best Height Fit. V tom může nastat případ, kdy lze předmět uložit na výšku i na šířku se stejným rozdílem mezi horní hranou předmětu a stropem police. U algoritmu Best Area Fit se tento shodný rozdíl vynásobí druhým rozměrem předmětu a tím dostaneme plochu, která uložením vznikne mezi předmětem a stropem police. Předmět je uložen tak, aby tato plocha byla co nejmenší.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

navysku – logická proměnná, pokud je předmět otočen při ukládání na výšku, má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

m – index police

p – počet polic

w – šířka předmětu

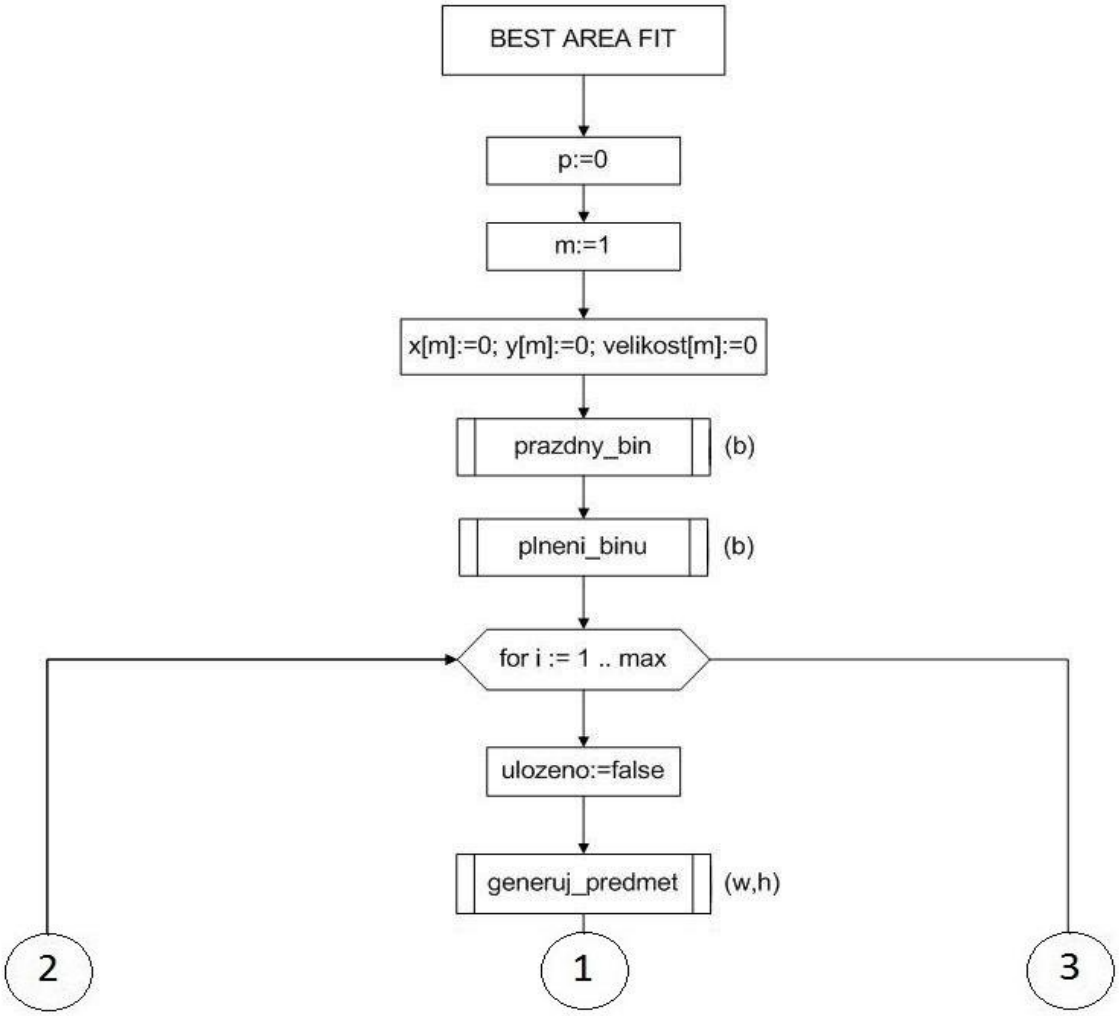
h – výška předmětu

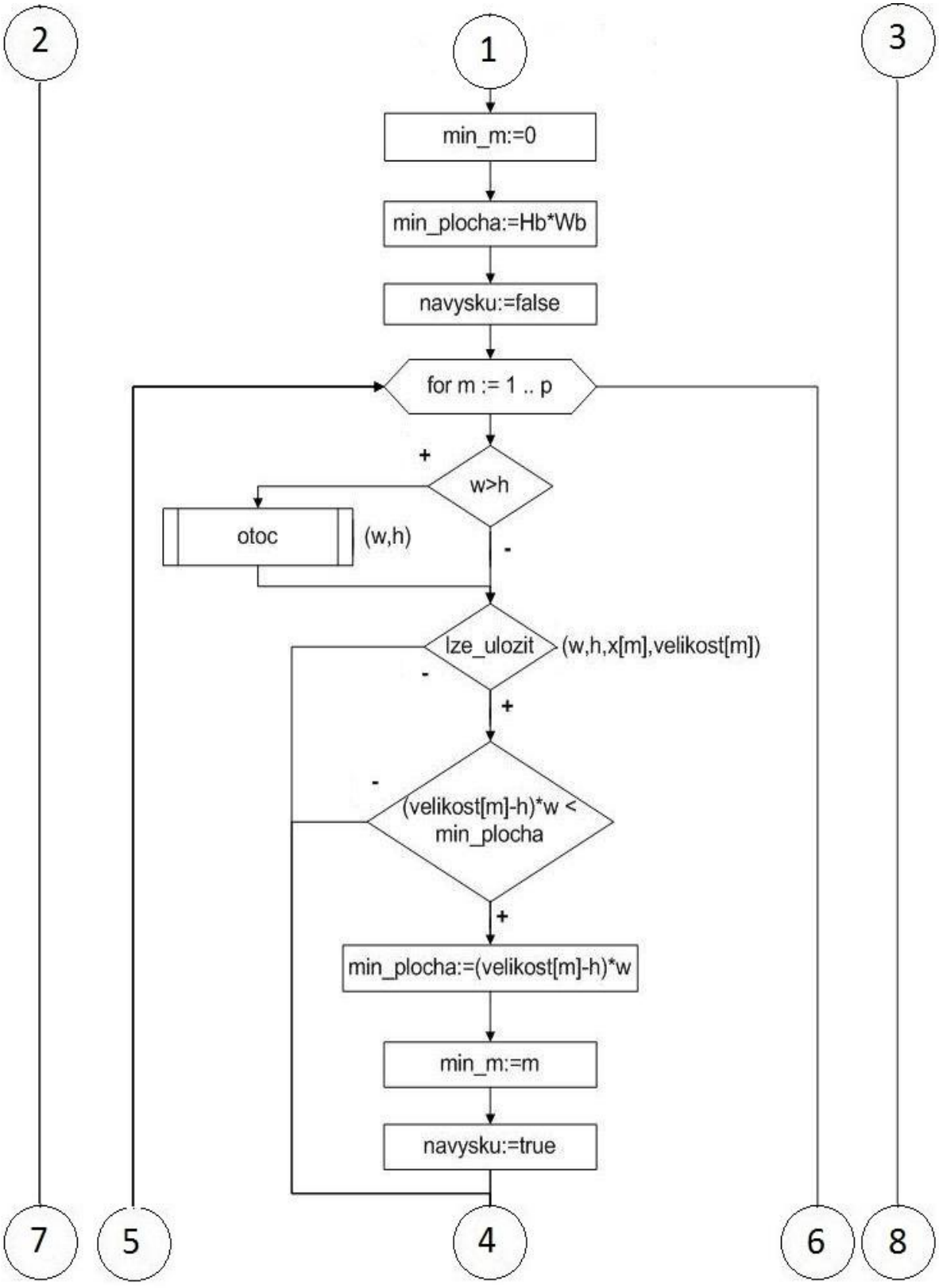
min_plocha – nejmenší plocha mezi horní hranou předmětu a stropem police

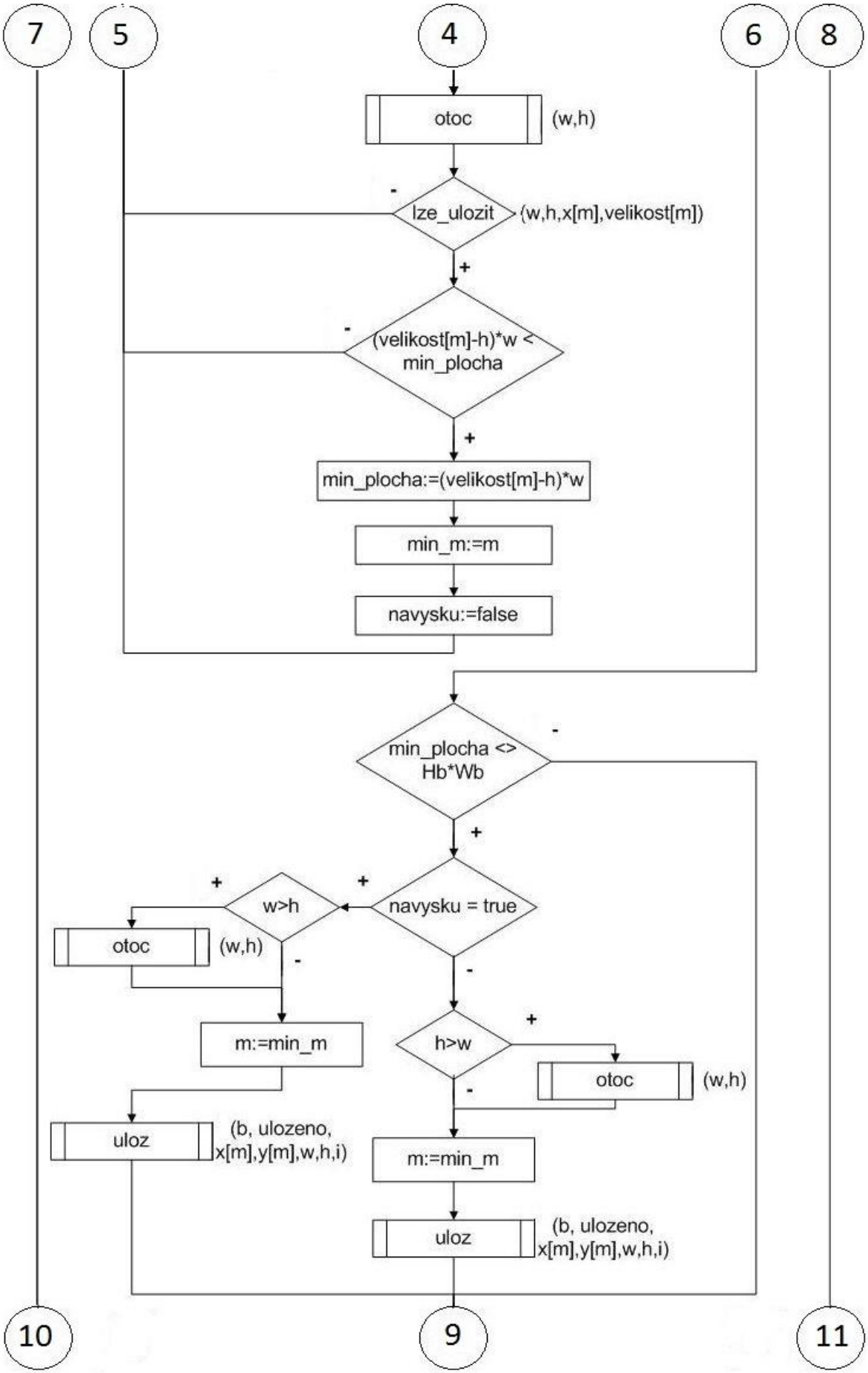
min_m – index police s nejmenším rozdílem

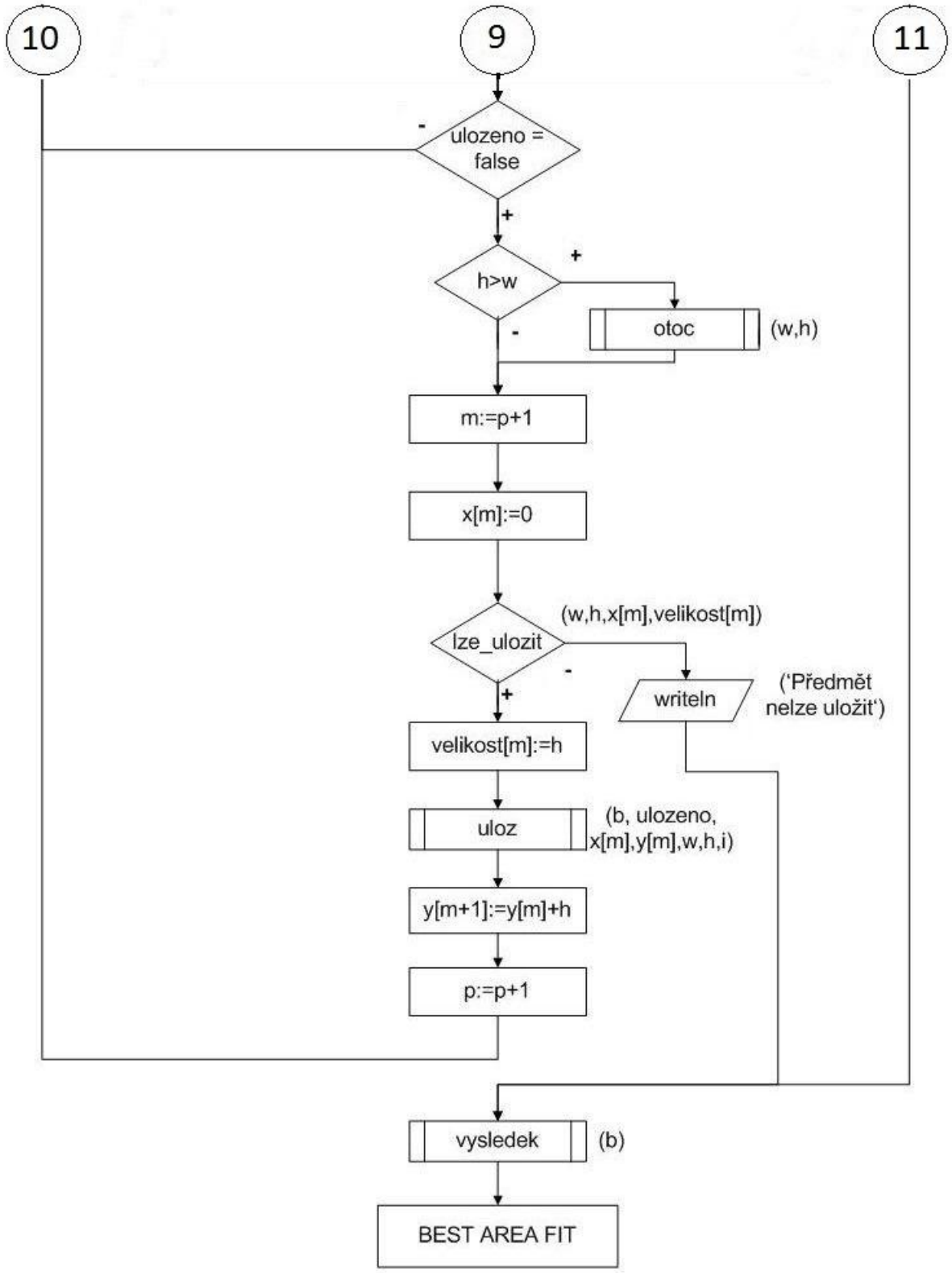
x – jednorozměrné pole zaplněnosti jednotlivých polic

y – jednorozměrné pole souřadnic spodků polic




2
1
3
min_m:=0
min_plocha:=Hb*Wb
navysku:=false
for m := 1 .. p
+
w>h
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
-
+
-
(velikost[m]-h)*w <
min_plocha
+
min_plocha:=(velikost[m]-h)*w
min_m:=m
navysku:=true
7
5
4
6
8


7
5
4
6
8
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
(velikost[m]-h)*w <
min_plocha
+
min_plocha:=(velikost[m]-h)*w
min_m:=m
navysku:=false
min_plocha <>
Hb*Wb
-
+
+
+
w>h
navysku = true
otoc
(w,h)
-
-
m:=min_m
h>w
+
otoc
(w,h)
-
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
m:=min_m
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
10
9
11



Obr. 4-12 Algoritmus Best Area Fit

4.2.6 Shelf Worst Width Fit

Na rozdíl od Best Width Fit algoritmu nehledá algoritmus nejmenší zbytek volného místa na jednotlivých policích, ale hledá naopak ten největší. Předmět je následně uložen do police

s maximálním zbývajícím místem na polici. Jedinou výjimkou u Worst algoritmů je nalezení police, která by uložením předmětů zcela zaplnila svou šířku, v takovém případě se předmět uloží do této police a zaplní ji.

Proměnné: ulozeno – logická proměnná, při uložení předmětu má hodnotu true

navysku – logická proměnná, pokud je předmět otočen při ukládání na výšku, má hodnotu true

b – kontejner (bin)

i – číslo předmětu (index)

m – index police

p – počet polic

w – šířka předmětu

h – výška předmětu

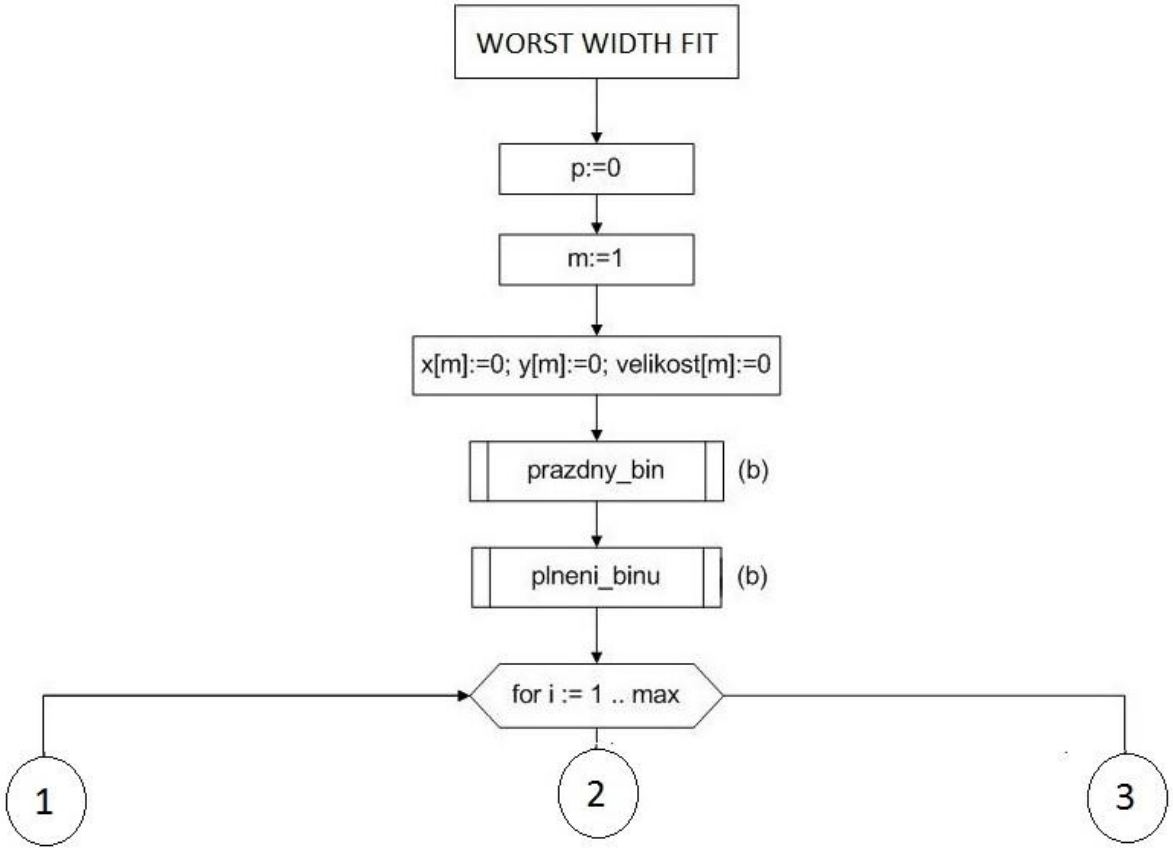
max_rozdil – největší zbývající volné místo na polici

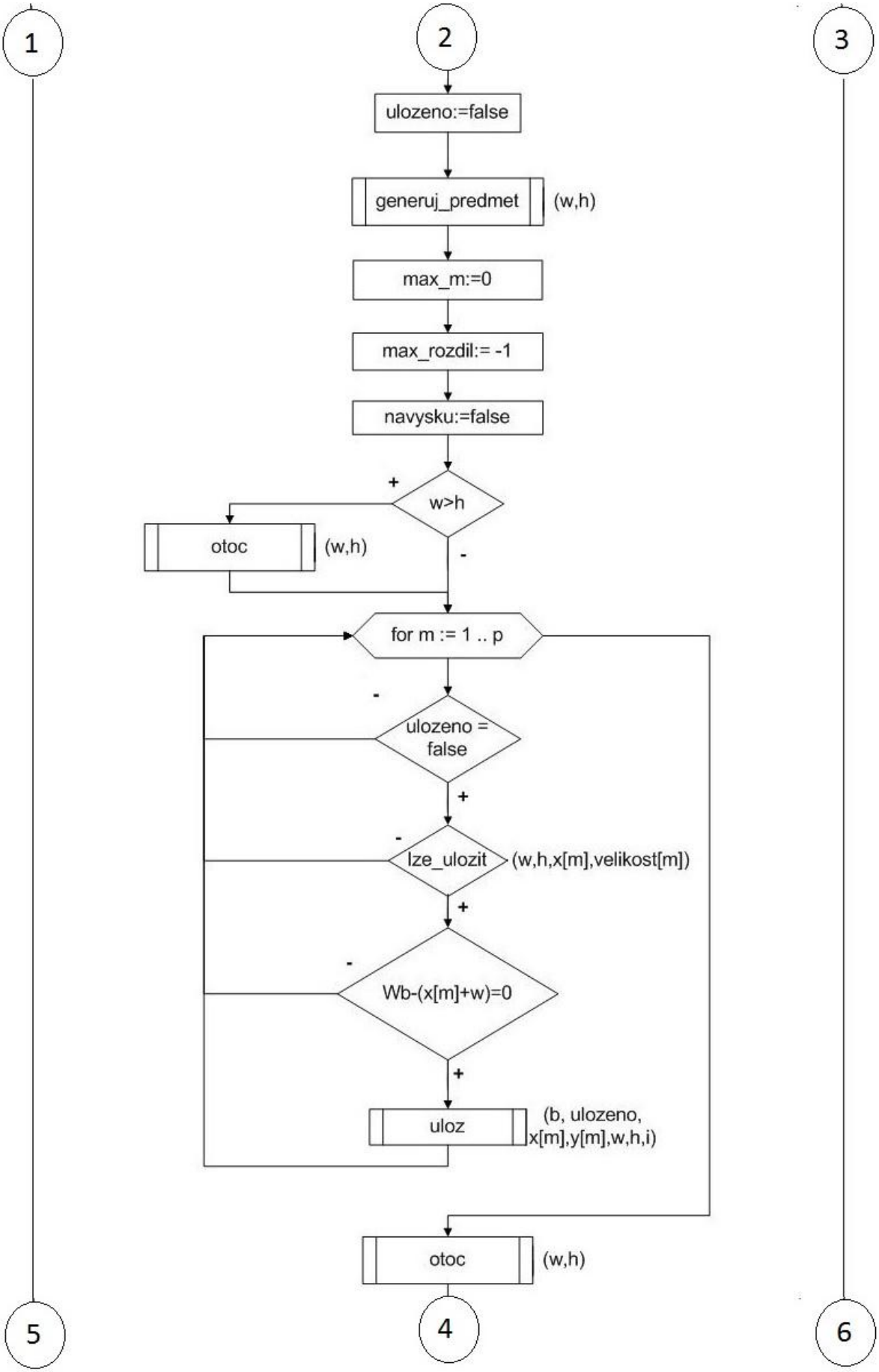
max_m - index police s největším zbývajícím volným místem

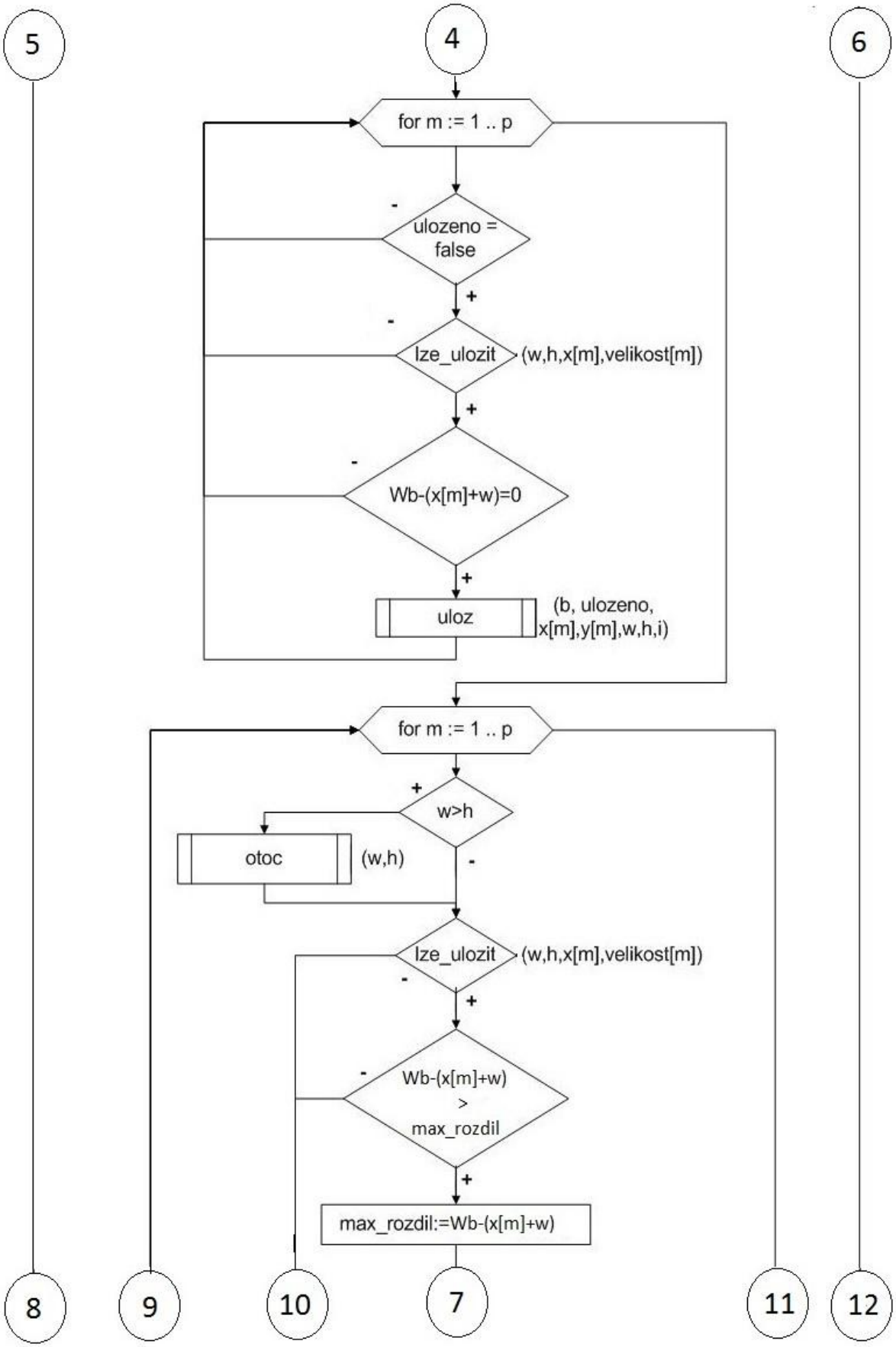
x – jednorozměrné pole zaplněnosti jednotlivých polic

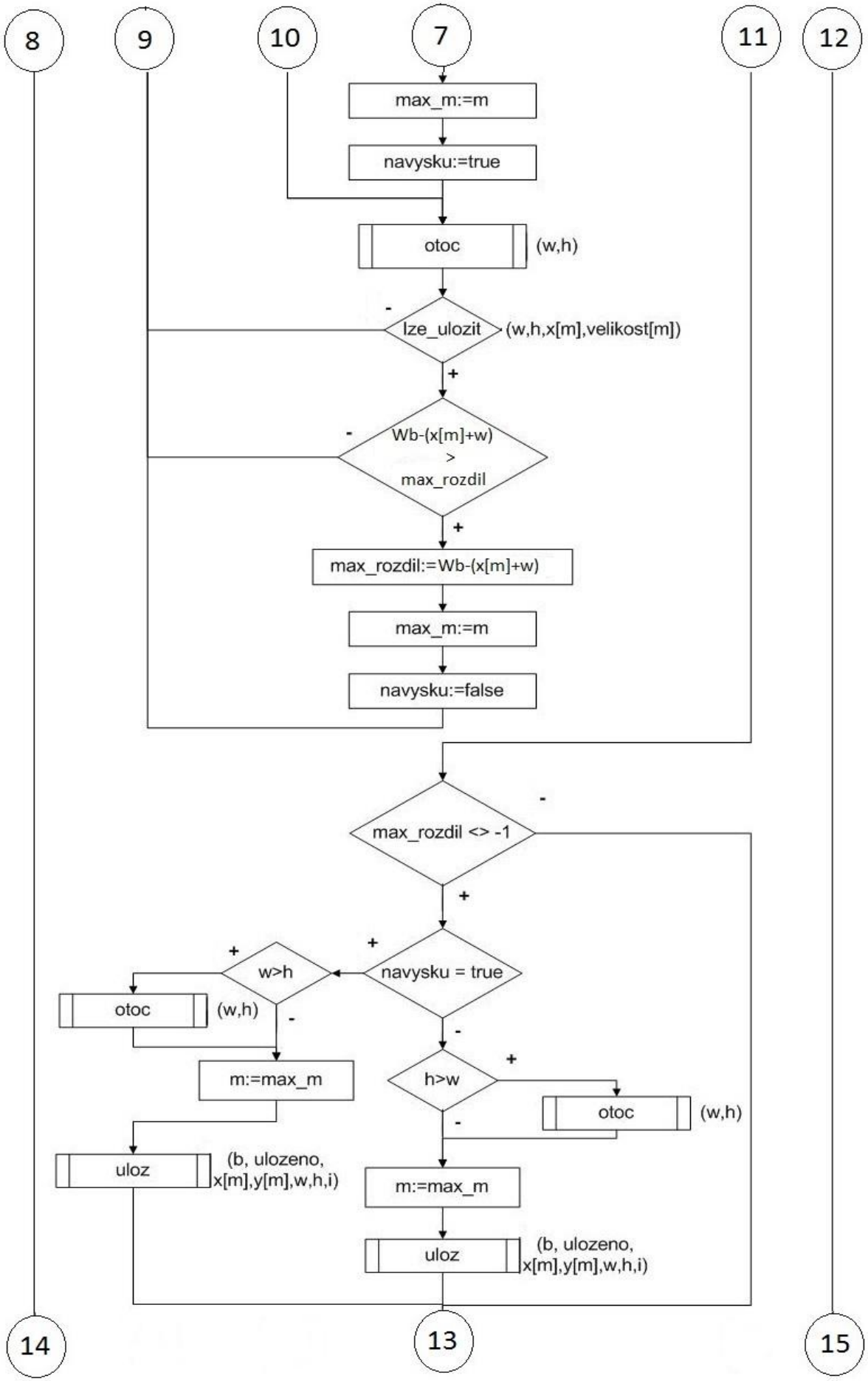
y – jednorozměrné pole souřadnic spodků polic

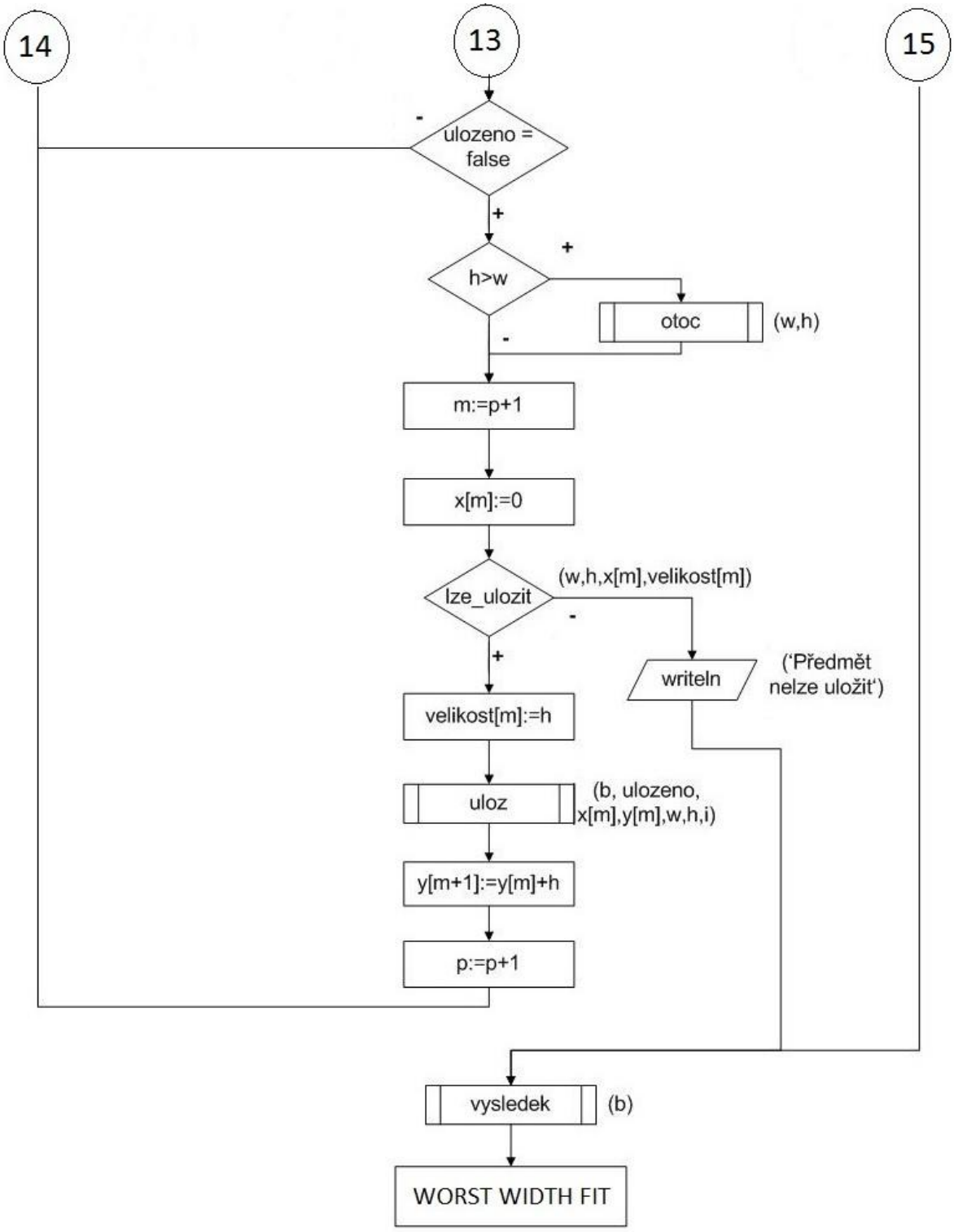
velikost – jednorozměrné pole velikostí (výšek) polic




1
2
3
ulozeno:=false
generuj_predmet
(w,h)
max_m:=0
max_rozdil:= -1
navysku:=false
+
w>h
otoc
(w,h)
-
for m := 1 .. p
-
ulozeno = false
+
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
Wb-(x[m]+w)=0
+
uloz
(b, ulozeno, x[m],y[m],w,h,i)
otoc
(w,h)
5
4
6


5
4
6
for m := 1 .. p
-
ulozeno = false
+
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
Wb-(x[m]+w)=0
+
uloz
(b, ulozeno, x[m],y[m],w,h,i)
for m := 1 .. p
+
w>h
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
-
+
-
Wb-(x[m]+w) > max_rozdil
+
max_rozdil:=Wb-(x[m]+w)
8
9
10
7
11
12


8
9
10
7
11
12
max_m:=m
navysku:=true
otoc
(w,h)
-
lze_ulozit
(w,h,x[m],velikost[m])
+
-
Wb-(x[m]+w)
>
max_rozdil
+
max_rozdil:=Wb-(x[m]+w)
max_m:=m
navysku:=false
max_rozdil <> -1
-
+
+
+
w>h
navysku = true
otoc
(w,h)
-
-
m:=max_m
h>w
+
otoc
(w,h)
-
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
m:=max_m
uloz
(b, ulozeno,
x[m],y[m],w,h,i)
14
13
15



Obr. 4-1 Algoritmus Worst Width Fit

4.2.7 Shelf Worst Height Fit

Na rozdíl od Best Height Fit algoritmu nehledá algoritmus nejmenší rozdíl mezi horní hranou předmětu a stropem police, ale hledá naopak ten největší. Předmět je následně uložen do police s maximálním rozdílem. Jedinou výjimkou u Worst algoritmů je nalezení police, která

by uložením předmětů zcela zaplnila svou šířku, v takovém případě se předmět uloží do této police a zaplní ji.

Proměnné a vývojový diagram jsou naprosto totožné jako u Worst Width Fit algoritmu s tím rozdílem, že do proměnné max_rozdil přiřazuje rozdíl (velikost[m]-h).

4.2.8 Shelf Worst Area Fit

Na rozdíl od Best Area Fit algoritmu nehledá algoritmus nejmenší plochu mezi horní hranou předmětu a stropem police, ale hledá naopak tu největší. Předmět je následně uložen do police s maximální touto plochou. Jedinou výjimkou u Worst algoritmů je nalezení police, která by uložením předmětů zcela zaplnila svou šířku, v takovém případě se předmět uloží do této police a zaplní ji.

Proměnné a vývojový diagram jsou naprosto totožné jako u Worst Width Fit algoritmu s tím rozdílem, že do proměnné max_rozdil přiřazuje součin (velikost[m]-h)*w.

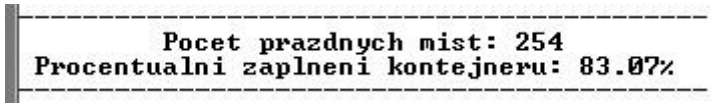
4.3 Implementace policových algoritmů do jazyku pascal

Algoritmy zmíněné v předchozí podkapitole byly implementovány pomocí programovacího jazyku Pascal.

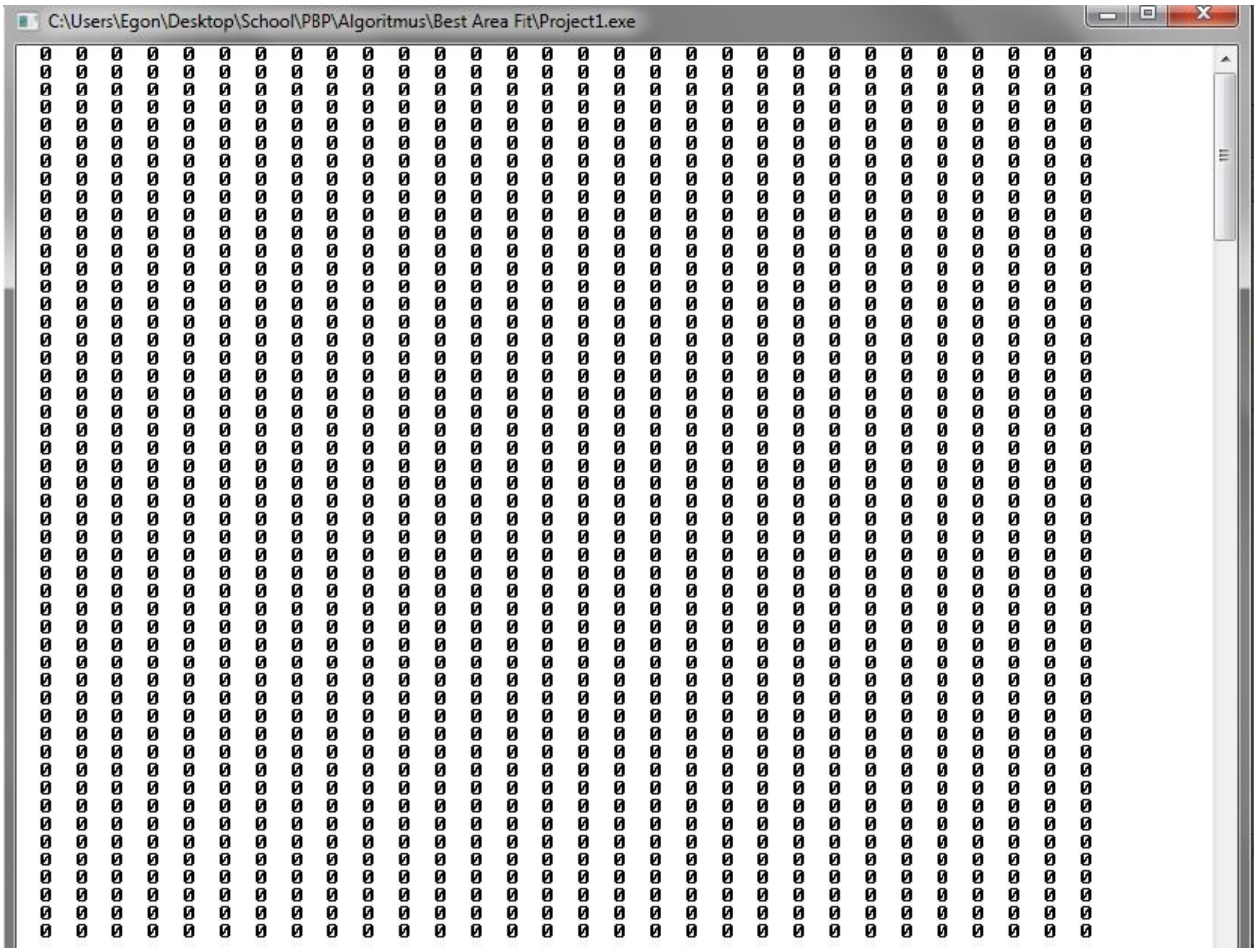
Jeden z vytvořených programů generoval předměty, ukládal je a zároveň seznam předmětů posílal ostatním algoritmům. Tímto způsobem bylo dosaženo měřitelných výsledků uložení stejných předmětů pomocí různých algoritmů.

Pro vlastní měření byla zvolena šířka kontejneru Wb = 120 a výška kontejneru Hb = 80 (rozměr europalety v centimetrech). Rozměry předmětů byly generovány náhodně mezi hodnotami 5 a 15. Při takto zvolených rozměrech dosáhlo průměrné zaplnění kontejneru nejvíce kolem 70%. Toto procento je dáno především zvolenými rozměry. Při zachování rozměrů předmětů a zvětšení rozměrů kontejneru by toto procento značně vzrostlo.

Algoritmus vždy po ukončení ukládání předmětů na obrazovku vypsal počet zbývajících volných míst a procentuální zaplnění kontejneru. Tato hodnota byla následně ještě zapsána do textového souboru. Z textových souborů všech algoritmů byly hodnoty vloženy do tabulky v excelu a následně byla spočtena průměrná hodnota těchto hodnot.



Obr. 4-14 Výsledek uložení



Obr. 4-15 Prazdný kontejner

1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	9	9	9	11	11	11	11	11	11	18	18	18	18	18	18	18	28	28	28	28	28	28	28	28
1	1	1	1	1	1	13	13	13	11	11	11	11	11	11	18	18	18	18	18	18	18	29	29	29	29	29	29	29	0
2	2	2	2	2	2	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
2	2	2	2	2	2	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
2	2	2	2	2	2	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
2	2	2	2	2	2	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
3	3	3	3	0	0	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
3	3	3	3	0	0	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
3	3	3	3	0	0	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
3	3	3	3	0	0	13	13	13	12	12	12	12	12	0	19	19	19	19	19	19	19	29	29	29	29	29	29	29	0
3	3	3	3	0	0	14	14	14	12	12	12	12	12	0	21	21	21	21	21	21	21	30	30	30	30	30	0	0	0
3	3	3	3	0	0	14	14	14	15	15	15	15	15	0	21	21	21	21	21	21	21	30	30	30	30	30	0	0	0
3	3	3	3	0	0	14	14	14	15	15	15	15	15	0	21	21	21	21	21	21	21	30	30	30	30	30	0	0	0
4	4	4	4	4	4	14	14	14	15	15	15	15	15	0	21	21	21	21	21	21	21	30	30	30	30	30	0	0	0
4	4	4	4	4	4	25	25	25	15	15	15	15	15	0	21	21	21	21	21	21	21	30	30	30	30	30	0	0	0
4	4	4	4	4	4	25	25	25	15	15	15	15	15	0	23	23	23	23	23	23	23	31	31	31	31	31	0	0	0
4	4	4	4	4	4	25	25	25	15	15	15	15	15	0	23	23	23	23	23	23	23	31	31	31	31	31	0	0	0
5	5	5	5	5	5	25	25	25	15	15	15	15	15	0	23	23	23	23	23	23	23	31	31	31	31	31	0	0	0
5	5	5	5	5	5	25	25	25	15	15	15	15	15	0	23	23	23	23	23	23	23	31	31	31	31	31	0	0	0
5	5	5	5	5	5	25	25	25	16	16	16	16	16	0	23	23	23	23	23	23	23	31	31	31	31	31	0	0	0
6	6	6	6	6	6	25	25	25	16	16	16	16	16	0	24	24	24	24	0	0	0	31	31	31	31	31	0	0	0
6	6	6	6	6	6	25	25	25	16	16	16	16	16	0	24	24	24	24	0	0	0	31	31	31	31	31	0	0	0
6	6	6	6	6	6	33	33	33	16	16	16	16	16	0	24	24	24	24	0	0	0	31	31	31	31	31	0	0	0
6	6	6	6	6	6	33	33	33	16	16	16	16	16	0	24	24	24	24	0	0	0	31	31	31	31	31	0	0	0
6	6	6	6	6	6	33	33	33	16	16	16	16	16	0	24	24	24	24	0	0	0	32	32	32	32	32	0	0	0
6	6	6	6	6	6	33	33	33	16	16	16	16	16	0	24	24	24	24	0	0	0	32	32	32	32	32	0	0	0
6	6	6	6	6	6	33	33	33	16	16	16	16	16	0	24	24	24	24	0	0	0	32	32	32	32	32	0	0	0
6	6	6	6	6	6	33	33	33	17	17	17	17	0	0	24	24	24	24	0	0	0	32	32	32	32	32	0	0	0
6	6	6	6	6	6	33	33	33	17	17	17	17	0	0	26	26	26	26	0	0	0	32	32	32	32	32	0	0	0
7	7	7	7	7	0	35	35	35	17	17	17	17	0	0	26	26	26	26	0	0	0	32	32	32	32	32	0	0	0
7	7	7	7	7	0	35	35	35	17	17	17	17	0	0	26	26	26	26	0	0	0	32	32	32	32	32	0	0	0
7	7	7	7	7	0	35	35	35	17	17	17	17	0	0	26	26	26	26	0	0	0	32	32	32	32	32	0	0	0
7	7	7	7	7	0	35	35	35	17	17	17	17	0	0	26	26	26	26	0	0	0	32	32	32	32	32	0	0	0
8	8	8	8	8	0	0	0	0	17	17	17	17	0	0	26	26	26	26	0	0	0	34	34	34	34	34	34	34	0
8	8	8	8	8	0	0	0	0	17	17	17	17	0	0	26	26	26	26	0	0	0	34	34	34	34	34	34	34	0
8	8	8	8	8	0	0	0	0	22	22	22	22	22	22	26	26	26	26	0	0	0	34	34	34	34	34	34	34	0
10	10	10	10	10	0	0	0	0	22	22	22	22	22	22	27	27	27	27	27	27	0	34	34	34	34	34	34	34	0
10	10	10	10	10	0	0	0	0	22	22	22	22	22	22	27	27	27	27	27	27	0	34	34	34	34	34	34	34	0
10	10	10	10	10	0	0	0	0	22	22	22	22	22	22	27	27	27	27	27	27	0	36	36	36	36	36	36	36	36
10	10	10	10	10	0	0	0	0	22	22	22	22	22	22	27	27	27	27	27	27	0	36	36	36	36	36	36	36	36
20	20	20	20	20	20	0	0	0	22	22	22	22	22	22	27	27	27	27	27	27	0	36	36	36	36	36	36	36	36
20	20	20	20	20	20	0	0	0	22	22	22	22	22	22	0	0	0	0	0	0	0	36	36	36	36	36	36	36	36
20	20	20	20	20	20	0	0	0	22	22	22	22	22	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Obr. 4-16 Uložené předměty v kontejneru

5 Závěr

Bylo provedeno 1000 ukládání vždy stejných předmětů všemi algoritmy, z výsledných hodnot procentuálních zaplnění jednotlivých ukládání byl spočten aritmetický průměr. Tyto průměrné hodnoty byly seřazeny sestupně od nejlepšího výsledku k nejhoršímu.

Průměrné zaplnění kontejneru jednotlivými algoritmy:

Best Area Fit:	72,78%
Best Height Fit:	71,43%
First Fit:	68,43%
Worst Width Fit:	68,32%
Best Width Fit:	58,30%
Worst Height Fit:	53,20%
Worst Area Fit:	53,19%
Next Fit:	44,59%

Nejlepšího průměrného výsledku ukládání bylo dosaženo Best Area Fit algoritmem, průměrné výsledky algoritmů Best Height Fit, First Fit a Worst Width Fit jsou horší pouze v řádu několika málo procent. Naopak Next Fit algoritmus dosáhl oproti Best Area Fit algoritmu o více než třetinu horšího průměrného výsledku.

Řádově byly ukládány desítky předmětů. Při zvýšení počtu předmětů lze předpokládat zvýšení procentuálního zaplnění kontejneru, ale také zvýšení rozdílů mezi průměrnými hodnotami jednotlivých algoritmů.

Pro praxi by nejlepší volbou bylo využít Best Area Fit algoritmus. Nastavit vlastní rozměr kontejneru a zadat rozměry předmětů, které mají být uloženy. Výstupem algoritmu by pak byl plán, jak předměty rozmístnit. U ukládání předmětů do kontejneru by se jednoho o půdorys uložení předmětů. U ukládání předmětů do regálů by se jednalo o nárys uložení předmětů. Uživatel by na plánu viděl, jaké má nastavit výšky jednotlivých polic, dále které předměty mají být uloženy do jednotlivých polic a v jakém pořadí.

Seznam obrázků

Seznam příloh

Seznam použitých zdrojů

1. **Lodi, Andrea.** *Algorithms for Two-Dimensional Bin Packing and Assignment Problems.* Padova : Universita Delgi Studi Di Bologna, 1999.
2. **Horký, Adam.** *Multi-agent solver for Multi-dimensional Bin Packing Problem - diplomová práce.* Praha : České vysoké učení technické v Praha, Fakulta elektrotechnicka, Katedra kybernetiky, 2012.
3. **Kotásek, Jakub.** *Plánování nakládky jako součást aplikace DCIx - diplomová práce.* Plzeň : Západočeská Univerzitra v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, 2013.
4. **Jylänki, Jukka.** http://clb.demon.fi. [Online] 27. Únor 2010. [Citace: 26. Září 2013.] http://clb.demon.fi/files/RectangleBinPack.pdf.
5. —. http://clb.demon.fi. [Online] 19. Září 2009. [Citace: 20. Říjen 2013.] http://clb.demon.fi/rectangle-bin-packing.
6. **Lodi, Andrea.** Recent advances on two-dimensional bin packing problems. *Discrete Applied Mathematics.* Elsevier, 2002, Sv. 123, 1-3.
7. **Wei, Lijun, Zhang, Defu a Chen, Qingshan.** A least wasted first heuristic algorithm for the rectangular packing problem. *Computers & Operations Research.* Elsevier, 2009, Sv. 36, 5.
8. **Virius, M.** *Základy programování — úvod do Turbo Pascalu.* Praha : ČVUT, 1999. ISBN 80-01-01553-X.

Příloha č.1

Ukázka hodnot měření

Index ukládání	BAF	BHF	BWF	FF	NF	WAF	WHF	WWF
1	68,84	68,84	61,56	61,56	43,27	61,56	61,56	61,56
2	70,24	70,24	54,05	70,24	42,02	45,52	45,52	62,21
3	76,38	66,26	70,2	76,38	49,36	49,36	49,36	76,38
4	70,48	70,48	52,56	70,48	52,56	64,06	64,06	70,48
5	50,26	50,26	52,3	50,26	50,26	50,26	50,26	52,3
6	63,39	63,39	54,16	65,43	47,3	54,16	54,16	63,39
7	68,33	68,33	53,45	68,33	40,55	53,45	53,45	64,48
8	68,71	68,71	66,94	68,71	49,5	49,5	49,5	68,71
9	83,01	83,01	55,27	55,27	41,03	55,27	55,27	55,27
10	79,79	79,79	57,27	71,1	34,63	50,05	50,05	71,1
11	79,71	79,71	53,17	79,71	44,15	41,03	41,03	64,21
12	74,19	74,19	57,33	68,17	48,27	57,33	57,33	74,19
13	77,08	77,08	74,33	78,23	45,47	45,47	45,47	69,57
14	71,15	68,63	49,78	68,63	49,78	71,15	71,15	71,15
15	56,9	58,5	38,49	58,5	38,49	47,06	47,06	56,9
16	64,52	64,52	59,23	66,15	56,58	59,75	59,75	66,15
17	78,82	78,82	57,98	80,43	55,11	55,11	55,11	80,43
18	76,58	75,54	60,42	60,42	37,59	51,08	51,08	76,58
19	80,87	42,35	42,35	42,35	42,35	42,35	42,35	77,52
20	67,51	68,97	55,61	68,97	39,82	55,61	55,61	68,97
21	75,89	77,01	54,36	62,08	33,48	44,99	44,99	70,73
22	70,6	70,6	62,62	70,6	48,74	46,99	46,99	70,6
23	84,78	84,78	55,35	69,74	55,35	55,35	55,35	79,31
24	87,92	87,92	52,38	51,16	40,61	52,38	52,38	52,38
25	62,49	62,49	46,76	62,49	54,16	54,16	54,16	64,09
26	63,22	63,22	61,39	64,98	61,39	53,46	53,46	63,22
27	74,4	74,4	68,31	67,56	48,82	59,77	59,77	75,12
28	75	75	65,64	75	48,9	53,63	53,63	67,78
29	78,14	79,39	63,8	78,14	40,85	54,94	54,94	79,39
30	76,4	76,4	61,77	75,27	39,74	61,77	61,77	72,43
31	57,4	57,4	47,45	57,4	45,41	43,42	43,42	57,4
32	85,33	85,33	53,72	57,89	44,79	53,72	53,72	57,89
33	72,57	74,03	57,17	66,52	43,48	57,17	57,17	74,03
34	65,96	65,96	67,33	67,33	50,01	55,35	55,35	67,33
35	71,91	71,91	55,84	71,91	39,94	38,19	38,19	71,91
36	54,8	54,8	54,8	56,06	54,8	55,55	55,55	56,06
37	74,04	74,04	61,25	74,04	45,5	45,5	45,5	66,27
38	80,15	81,6	46,08	67,16	46,08	46,08	46,08	67,16
39	61,62	61,62	46,7	63,11	37,72	46,7	46,7	63,11
40	63,35	63,35	59,54	63,35	59,54	58,19	58,19	64,19
41	83,07	69,95	65,96	69,95	46,95	46,95	46,95	59,58
42	84,97	85,62	40,95	78,5	40,95	40,95	40,95	68,31
43	82,72	82,72	64,2	81,69	48,81	48,81	48,81	48,81

44	66,43	66,43	66,43	66,43	43,54	66,43	66,43	66,43
45	56,56	56,56	53,85	56,56	43,02	56,56	56,56	58,32
46	69,7	71,32	45,77	71,32	41,95	45,77	45,77	71,32
47	73,21	73,21	57,44	74,52	34,35	57,44	57,44	57,44
48	78,91	78,91	58,4	80,13	35,39	57,25	57,25	78,91
49	84,17	84,17	45,26	74,62	36,71	55,74	55,74	79,65
50	77,33	77,33	74,41	77,33	44,29	65,91	65,91	77,33
51	63,65	63,65	65,25	63,65	42,32	62,4	62,4	63,65
52	78,53	78,53	64,76	70,09	38,59	53,28	53,28	78,53
53	68,36	68,36	65,05	68,36	31	54,98	54,98	68,36
54	76,59	76,59	54,81	77,62	47,31	47,31	47,31	57,97
55	87	79,29	53,73	69,9	36,48	41,22	41,22	87
56	78,24	79,61	51,2	51,2	36,58	51,2	51,2	51,2
57	86,24	82,61	56,18	81,18	39,4	42,94	42,94	56,18
58	78,35	78,35	61,26	79,39	35,78	57,19	57,19	78,35
59	66,01	55,06	52,66	55,06	47,35	52,66	52,66	55,06
60	74,05	74,05	51,26	75,95	44,54	51,26	51,26	72,68
61	67,92	67,92	51,81	67,92	45,9	46,93	46,93	69,37
62	80,15	80,15	67,22	67,22	38,34	58,58	58,58	78,77
63	74,16	74,16	62,07	63,17	27,33	60,82	60,82	74,16
64	57,1	57,1	57,1	57,1	48,23	52,15	52,15	57,1
65	78,45	78,45	62,84	63,88	37,49	48,96	48,96	63,88
66	74,05	74,05	49,52	75,43	40,17	45,79	45,79	74,05
67	82,78	82,78	71,46	72,86	46,27	46,27	46,27	72,86
68	77,5	79,1	62,69	62,69	45,17	55,66	55,66	77,5
69	82,14	82,14	56,78	83,63	40,75	55,64	55,64	62,06
70	83,09	84,26	60,58	60,58	46	44,25	44,25	60,58
71	67,43	67,43	59,62	67,43	36,03	59,62	59,62	59,62
72	73,42	74,77	50,94	61,35	44,82	62,61	62,61	61,35
73	69,72	69,72	52,81	52,81	39,86	70,47	70,47	52,81
74	63,29	64,33	64,33	63,29	45,69	55,17	55,17	64,33
75	74,53	74,53	42,32	74,53	36,72	42,32	42,32	64,82
76	47,22	47,22	45,72	48,98	36,89	47,22	47,22	47,22
77	79,6	79,6	61,79	66,92	50,81	64,64	64,64	61,79
78	82,21	55,76	55,76	55,76	36,51	52,78	52,78	55,76
79	39,65	39,65	41,14	39,65	41,14	41,14	41,14	39,65
80	67,95	67,95	67,03	67,95	51,29	63,14	63,14	67,95
81	73,17	73,17	54,3	74,92	54,3	54,3	54,3	74,92
82	75,05	75,05	62,71	75,05	45,51	50,62	50,62	75,49
83	71,32	69,22	41,75	69,22	53,54	41,75	41,75	65,89
84	73,77	73,77	61,48	67,58	51,26	55,32	55,32	73,77
85	69,7	69,7	66,06	69,7	56,76	42,01	42,01	69,7
86	79,37	73,5	51,57	78,21	57,47	51,57	51,57	78,68
87	59,13	59,13	53,91	59,13	45,1	53,91	53,91	60,73
88	62,34	62,34	56,52	62,34	32,82	56,52	56,52	63,83

89	85,49	85,49	64,11	70,71	35,9	50,39	50,39	70,71
90	74,91	74,91	50,92	65,2	57,26	47,34	47,34	57,26
91	67,83	67,83	36,73	67,83	49,23	56,26	56,26	67,83
92	66,81	66,81	68,27	66,81	42,71	63,4	63,4	68,27
93	58,39	58,39	49,91	55,16	43,27	55,16	55,16	58,39
94	80,36	81,04	51,18	58,71	47,9	51,18	51,18	77,93
95	76,25	76,25	45,6	45,6	45,6	45,6	45,6	45,6
96	82,71	82,71	39,62	81,87	39,62	39,62	39,62	78,68
97	84,76	64,51	73,61	80,28	44,42	55,21	55,21	64,51
98	81,64	77,96	57,6	83,01	42,17	37,53	37,53	77,7
99	75,18	76,8	67,56	65,06	53,47	59,52	59,52	65,06
100	68,78	68,78	68,78	68,78	44,23	44,23	44,23	68,78

Ukázka z měření procentuálního zaplnění kontejneru. Ukládány byly vždy stejné předměty všemi algoritmy.