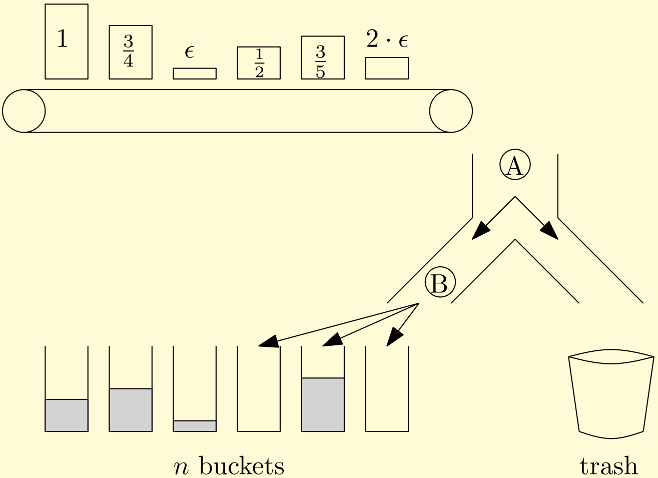
Online Knapsack

Maciej Pacut
joint work with
Marcin Bieńkowski, Tadeusz Dul, Krzysztof Piecuch

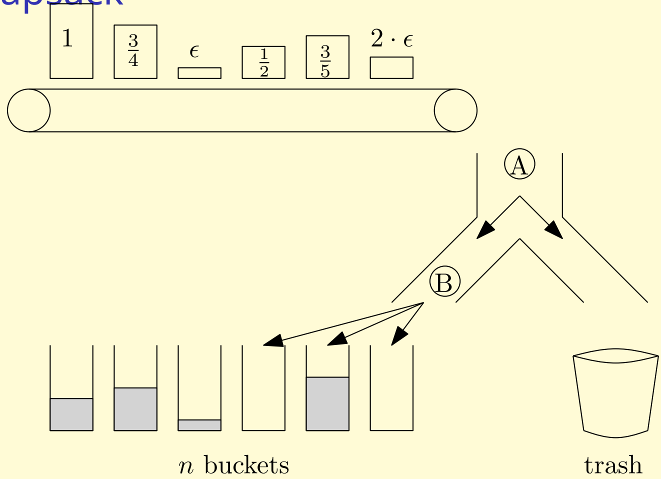
17 kwietnia 2020

Wariant Non-removable Proportional Online Knapsack



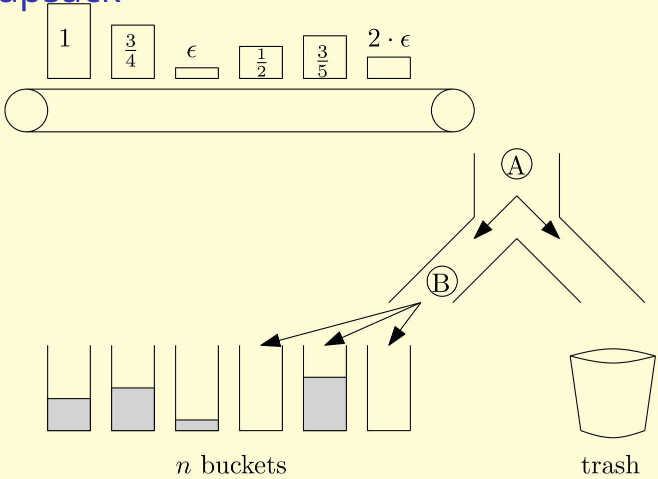
n kubełków o pojemności 1, sekwencja przedmiotów w trybie online, nie usuwamy ani nie przenosimy

Wariant Non-removable Proportional Online Knapsack



1. maksymalizacja sumy rozmiarów przyjętych przedmiotów
2. konkurencyjność: $\frac{ALG}{OPT_{OFF}}$

Wariant Non-removable Proportional Online Knapsack



1. Greedy FirstFit jest $1/2$-konkurencyjny
2. Górna granica konkurencyjności dla dowolnego algorytmu zrandomizowanego: $R = 1/(1 + \ln(2)) \approx 0.59$

Poprzednie rezultaty (Łukasz Jeż, Marek Cygan)

var	obj	case	deterministic lower	deterministic upper	randomized (oblivious) lower	randomized (oblivious) upper
R	1	gen.	∞	–	$\frac{e+1}{e} \approx 1.37$ [Thm. 7]	2 [Sec. 3.1]
		prop.	$\phi \approx 1.62$	$\phi \approx 1.62$	1.25 [Thm. 8]	$\phi \approx 1.62$
		unit	1	1	1	1
	max	gen.	$\phi \approx 1.62$ [Thm. 6]	2 [Thm. 3]	1	2
		prop.	≈ 1.38 [8]	≈ 1.38 [8]	1	≈ 1.38
		unit	1	1	1	1
	$\sum$	gen.	$\frac{7}{6} \approx 1.17$	$3 + O(\frac{1}{k})$ [Thm. 2]	$\frac{7}{6} \approx 1.17$	3 [Thm. 2]
		prop.	$\frac{8}{7} \approx 1.14$	1.5 [16]	$\frac{8}{7} \approx 1.14$ [Thm. 4]	2
		unit	$\frac{7}{6} \approx 1.17$	$3 + O(\frac{1}{k})$	$\frac{7}{6} \approx 1.17$ [1]	3
NR	1	gen.	∞	–	∞	–
		prop.	∞	–	2 [4]	2 [4]
		unit	∞	–	∞	–
	max	gen.	∞	–	∞	–
		prop.	2	2 [4]	2 [Thm. 12]	2
		unit	∞	–	∞ [Thm. 11]	–
	$\sum$	gen.	∞	–	∞	–
		prop.	$1 + \ln 2 \approx 1.69$	2 [Thm. 10]	$1 + \ln 2$ [Thm. 13]	2
		unit	∞	–	∞ [Thm. 11]	–

Temat dzisiejszej prezentacji

Notacja:

1. Duże przedmioty: $(1/2, 1]$ – mieszczą się po 1 w kubełku
2. Średnie przedmioty: $(1/3, 1/2]$ – mieszczą się po 2 w kubełku

Przedstawię dwa algorytmy:

1. Optymalny algorytm dla dużych przedmiotów
2. Optymalny algorytm dla dużych i średnich przedmiotów

Algorytm dla dużych przedmiotów $(1/2, 1]$

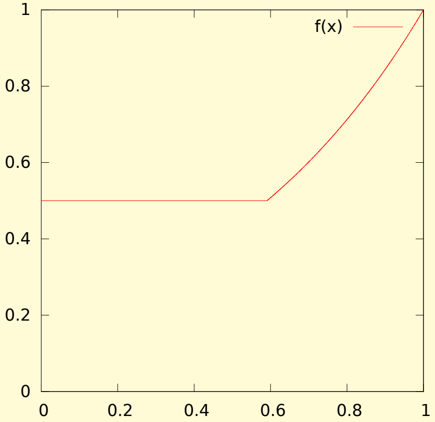
$$f(x) = \begin{cases} 1/2 & \text{if } x \leq R\ , \\ (2e)^{x-1} & \text{otherwise.} \end{cases}$$

Algorithm 1: ALG$_L$

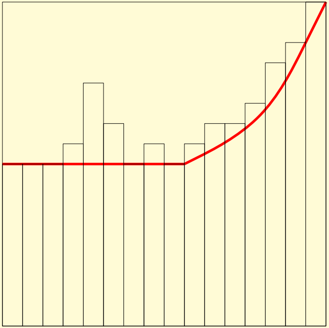
When new item of size s arrives:
b := number of filled buckets
if *b* = *n* **then**
  terminate
if $s \leq f((b+1)/n)$ **then**
  reject
else
  put item into any empty bucket

Wykres funkcji f

$$f(x) = \begin{cases} 1/2 & \text{if } x \leq R\ , \\ (2e)^{x-1} & \text{otherwise.} \end{cases}$$

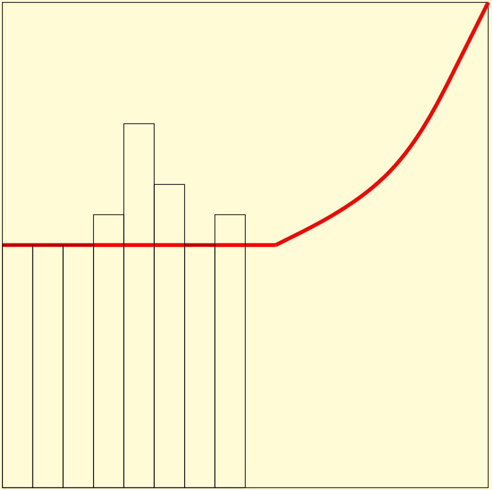


Zobrazowanie kubełków i funkcji progowej



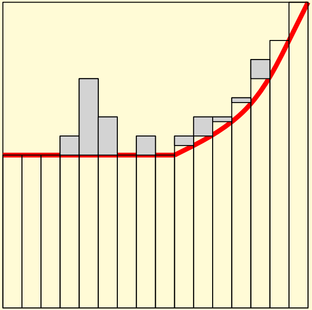
1. Próg dla nowego przedmiotu zależy od liczby przyjętych przedmiotów (nie od ich rozmiarów)
2. (metakomentarz) Stajemy się bardziej konserwatywni gdy mamy mniej zasobów

Analiza algorytmu dla dużych przedmiotów $(1/2, 1]$



Jeśli algorytm zakończył się zapełniając mniej niż $R \cdot n$ kubełków to zaakceptowaliśmy wszystko i jesteśmy optymalni

Analiza algorytmu dla dużych przedmiotów $(1/2, 1]$



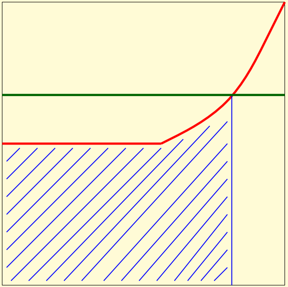
$$W = \sum \square$$

Jeśli algorytm zakończył zapełniając $b \geq R \cdot n$ kubełków to:

- adwersarz “zostawia OPTowi na koniec” przedmioty poniżej progu akceptacji $f(b)$
- $OPT \leq n \cdot f(b) + W$
- $ALG \geq n \cdot \int^b f(x)dx + W$

Pewna własność f

$$\forall_{b\in(R,1]} \frac{\int^b f(x)dx}{f(b)} = R$$



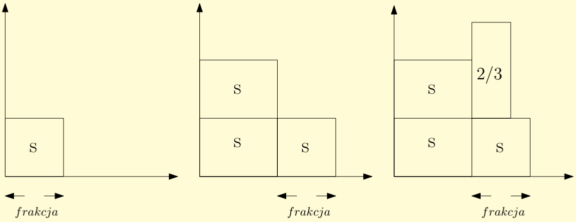
$$\Rightarrow \frac{ALG}{OPT} \geq \frac{\int^b f(x)dx + W}{f(b) + W} \geq R$$

Rozgrzewka – wszystkie średnie przedmioty mają tę samą wielkość

Algorytm dla dużych przedmiotów $(1/2, 1]$ i średnich przedmiotów o jednym, ustalonym rozmiarze s:

1. Traktujemy duże przedmioty jak Algorytm dla dużych przedmiotów
2. Akceptujemy wszystkie średnie przedmioty
3. Decyzja do podjęcia: średnie po 1 czy po 2 w kubełku? Odpowiedź: utrzymuj pewną frakcję (wszystkich kubełków) pojedynczych a po przekroczeniu progu: podwójne
4. Jeśli możemy to łączymy duże i średnie przedmioty

Rozgrzewka – wszystkie średnie przedmioty mają tę samą wielkość



1. Utrzymuj pewną frakcję (wszystkich kubełków) pojedynczych a po przekroczeniu progu: podwójne
2. Jeśli możemy to łączymy duże i średnie przedmioty

Analiza algorytmu – przypadek gdy zostały puste kubełki

Korzystamy z dwóch własności algorytmu:

1. Traktujemy duże przedmioty jak Algorytm 1
2. Akceptujemy wszystkie średnie przedmioty

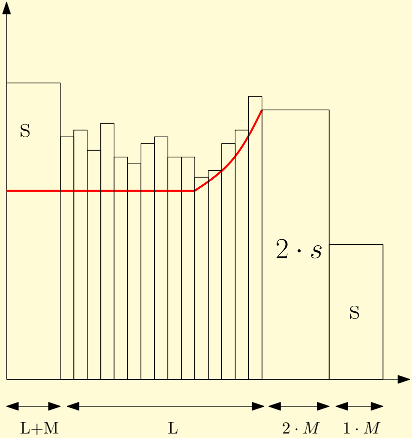
Szacujemy zysk OPTa i Algorytmu 2:

1. $ALG(I) = ALG_L(I_L) + weight(I_M)$
2. $OPT(I) \leq OPT(I_L) + OPT(I_M) \leq OPT(I_L) + weight(I_M)$

Sprowadzamy analizę Algorytmu 2 do analizy Algorytmu 1:

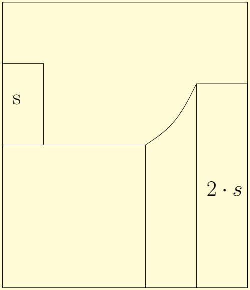
$$\frac{ALG(I)}{OPT(I)} \geq \frac{ALG_L(I_L) + weight(I_M)}{OPT(I_L) + weight(I_M)} \geq \frac{ALG_L(I_L)}{OPT(I_L)}$$

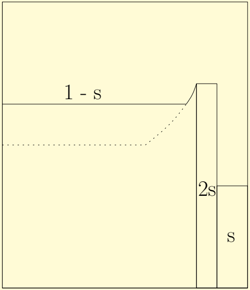
Analiza algorytmu – przypadek, gdy wszystkie kubełki zajęte



1. Rozpatrujemy sytuację po zakończeniu działania algorytmu
2. Szacujemy zysk na każdej klasie kubełków
3. Dwa podprzypadki: istnieje klasa $1M$ i nie istnieje

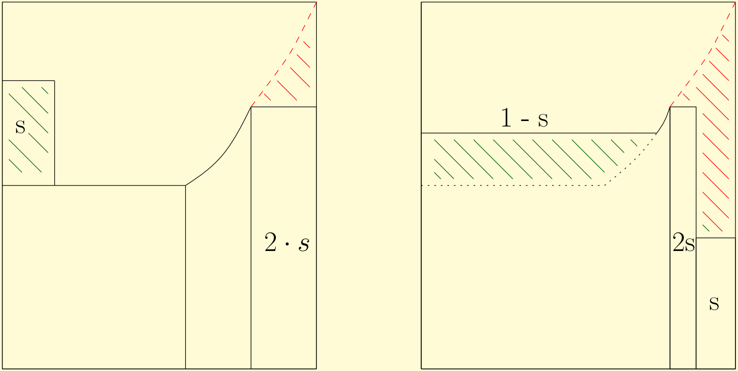
Analiza algorytmu – przypadek, gdy wszystkie kubełki zajęte





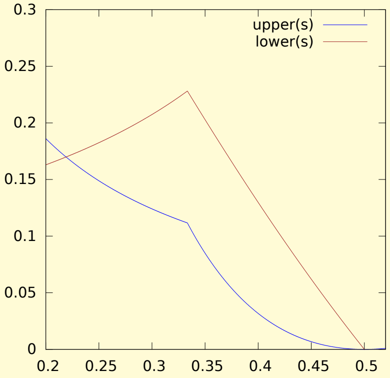
- (po lewej) nie zostały kubełki pojedyncze – były, ale zostały zjedzone przez duże
- (po prawej) zostały kubełki pojedyncze – nie mogliśmy połączyć ich z dużymi, więc szacujemy duże przez max$\{f(b_i), 1 - s\}$

Analiza algorytmu – dwa przypadki i ograniczenia na frakcję



Po lewej: chcemy, żeby frakcja pojedynczych była jak największa. Po prawej: chcemy, żeby frakcja była jak najmniejsza.

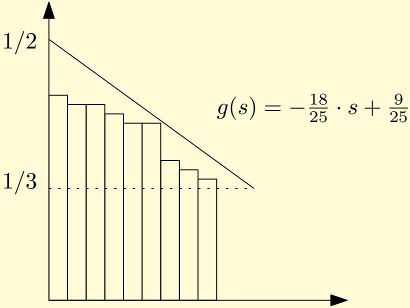
Analiza algorytmu – dwa przypadki i ograniczenia na frakcję



Nieciągłość pochodnej w $1/3$ jest spowodowana faktem, że analizujemy wkładanie 2 średnich przedmiotów do kubełka, a mniejsze przedmioty wchodzą po $3, 4, \ldots$.

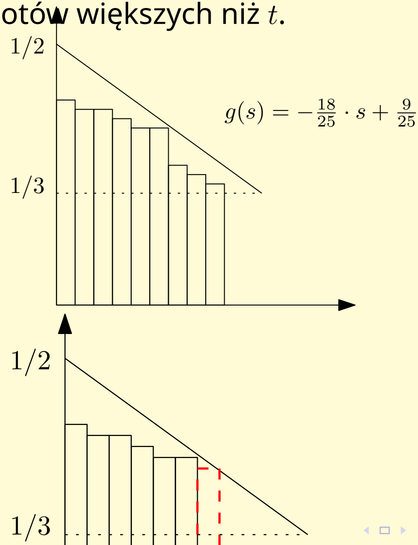
Więcej rozmiarów średnich przedmiotów

1. Definiujemy strukturę, w której trzymamy wszystkie średnie przedmioty POZA tymi po dwa średnie w kubełku.
2. W strukturze są zarówno przedmioty pojedyncze jak i połączone z dużymi.
3. Przedmioty mieszczą się w strukturze jeśli posortowane znajdują się pod wykresem funkcji g
4. Funkcja $g^{-1}(s)$ określa, ile przedmiotów $\geq s$ utrzymujemy pojedynczo. Funkcja g^{-1} musi zawierać się między ograniczeniami na frakcję.

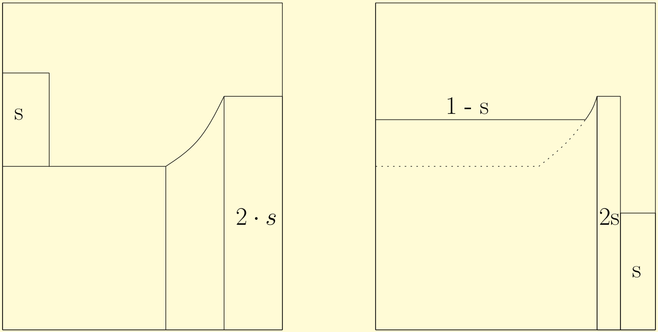


Relacja minimalny ciasny-minimalny odrzucony ze struktury

Mówimy, że t jest ciasny jeśli nie możemy dołożyć kolejnego przedmiotu o wielkości t do struktury z powodu zbyt dużej liczby przedmiotów większych niż t.



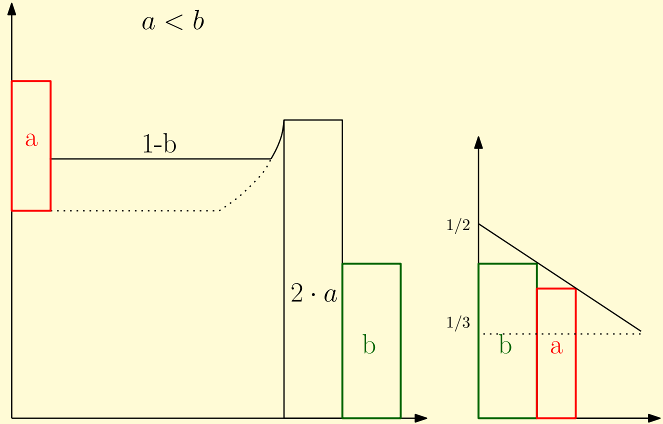
Więcej rozmiarów średnich przedmiotów



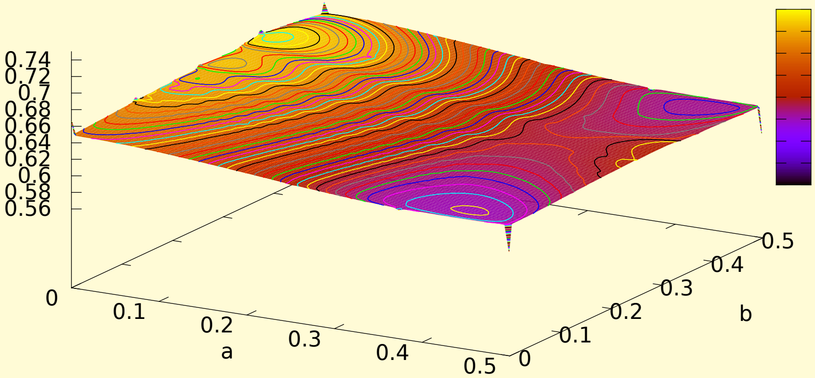
Jakie są argumenty funkcji, czyli czym jest s?
Po lewej: s to minimalny ciasny przedmiot
Po prawej: s to minimalny przedmiot

Nowy przypadek w analizie

Minimalny ciasny może być różny niż minimalny pojedynczy.
Wtedy a – minimalny ciasny, b – minimalny pojedynczy.



Minimalizacja funkcji dwuargumentowej



Wykres dla $a, b \in (0, 1/2)$, choć w $0.2103\ldots$ wykres przecina się z ograniczeniem dolnym.

Algorytm dla $(1/3, 1]$

When new item x arrives:
if *there are no empty buckets left* **then**
  terminate
if x *is large* **then**
  if $x \geq f((buckets(L) + buckets(L^+) + 1)/n)$ **then**
    if *there exists any M-bucket with* x *space left* **then**
      put x in the smallest bucket with x space left $(M \to L^+)$
    else
      put x into any empty bucket $(\epsilon \to L)$
  else
    reject
if *x is medium* **then**
  if $D \cup \{x\}$ *satisfies* D*-invariant* **then**
    if *there exists a L-containing bucket with* x *space left* **then**
      put x in this bucket (L or $L^+ \to L^+$)
    else
      put x into any empty bucket $(\epsilon \to M)$
  else
    if *there exist a* M_i*-bucket with* x *space left* **then**
      put x in this bucket $(M_* \to M_*)$
    else
      put x in an empty bucket and apply M_i label $(\epsilon \to M_*)$