# Tema Algoritmi Avansati

Eduard-Valentin Dumitrescul, grupa 232

March 6, 2024

## 1 Algoritmi Aproximativi

### 1.1 Knapsack

Fie S un șir de numere naturale $s_1, s_2, ..., s_n$ și K un număr natural, cu $K \geq s_i$ pentru orice i între 1 și n.

a) Scrieți un algoritm pseudo-polinomial care găsește suma maximă, dar care să fie $\leq K$, ce poate fi formată din elementele din S (numere întregi, pozitive, luate cel mult o singură dată). Indicați complexitatea de timp/spațiu a algoritmului propus de voi și justificați de ce acesta este corect (de ce soluția găsită este optimă). (1p)

b) Scrieți un algoritm aproximativ care calculează o sumă cel puțin pe jumătate de mare ca cea optimă, dar rulează în timp O(n) și complexitate spațiu O(1). (1p)

**Rezolvare: a)**

Complexitate: O(N*K) timp si O(K) spatiu

```
sum_reached[0] = 1    #suma 0 se obtine in mod implicit
for i=1 to k:
  sum_reached[i] = 0 #marcam faptul ca inca nu am obtinut suma i

#luam, pe rand, fiecare valoare din sirul S si incercam sa obtinem noi sume pe
   ↪ baza celor deja obtinute
for i=1 to n:
  for sum=k to 0:
    if sum_reached[sum] = 1 and i + sum <= k:
      sum_reached[i+sum] = 1

maximum_sum = arg(max(i | i∈{0,1,...,k}, sum_reached[i] = 1))
```

Stim ca suma 0 se poate obtine mereu (nealegand niciun numar din sir).

Presupunem ca am gasit toate sumele $\leq K$ ce pot fi obtinute adunand numere din $s_1, s_2, ..., s_{i-1}$. Acum, orice suma ce il contine pe $s_i$ si este formata cu numerele $s_1, s_2, ..., s_i$ este de forma $s_i + S_{i-1}$ , unde S este o suma obtinuta din numerele $s_1, s_2, ..., s_{i-1}$. Stiind ca am gasit toate astfel de $S_{i-1}$ rezulta ca vom gasi si toate sumele de forma $S_i$. Inductiv, vom gasi toate sumele $S_n \leq K$ posibile formate din $s_1, s_2, ..., s_n$.

**Rezolvare: b)**

Complexitate: O(N) timp si O(1) spatiu

```
Fie S lista de numere

sum = 0
maxValue = 0
for each s in S:
  if sum + s <= K:
    sum += s
  if max < s:
    maxValue = s

if maxValue >= K/2:
  ALG() = maxValue
else:
  ALG() = sum
```

OPT = valoarea solutiei optime
vmax = valoarea maxima din S
Daca $vmax \geq OPT/2$, atunci $ALG = vmax \geq OPT/2$, deci $ALG \geq 1/2 * OPT$.
Daca $vmax < OPT/2$, atunci fiecare numar din S este $\leq OPT/2$. Daca algoritmul va insuma toate numerele, atunci obtine solutia optima. Altfel, fie primul numar $s_i$ pe care nu il aduna si $sum$, suma obtinuta pana in acel punct.
$sum + s_i > K \geq OPT \Rightarrow sum > OPT - s_i$ (1)
$s_i < OPT/2 \Rightarrow OPT - s_i > OPT/2$ (2)
$(1) si (2) \Rightarrow sum > OPT/2$
$ALG \geq sum \Rightarrow ALG > OPT/2$
Asadar, ALG ete 1/2-aproximativ.

**Exemplu in care $ALG \approx OPT/2$:**

$S = \{10, 1, 2, 3, 4\}, K = 20$
$ALG = 10$
$OPT = 20$
$ALG/OPT = 2$

## 1.2 Load-Balancing - 2

Fie $ALG_1$ și $ALG_2$ doi algoritmi de rezolvare pentru aceeași problemă de minimizare. $ALG_1$ este un algoritm 2-aproximativ, respectiv $ALG_2$ este un algoritm 4-aproximativ. Stabiliți valoarea de adevăr a următoarelor propoziții, dând și o scurtă justificare:

a) Există cu siguranță un input I pentru care $ALG_2(I) \geq 2 \cdot ALG_1(I)$ (0,5p)
b) Nu există niciun input I pentru care $ALG_1(I) \geq 2 \cdot ALG_2(I)$ (0,5p)

**Rezolvare a)**

Nu este adevarat. Daca pentru un anume I, amandoi algoritmii dau solutia optima, relatia nu este adevarata.

**Rezolvare b)**

Nu este adevarat.
$ALG_1$ este 2-aproximativ $\Rightarrow ALG_1 \leq 2 \cdot OPT$
Presupunem ca exista I pentru care $ALG_1 \geq 2 \cdot ALG_2$, deci
$2 \cdot OPT \geq ALG_1 \geq 2 \cdot ALG_2$, de unde
$OPT \geq ALG_2$.
Asadar, pentru $ALG_2 = OPT$ si $ALG_1(I) = 2 \cdot OPT(I)$ inegalitatea are loc.

## 1.3 Load-Balancing 3

Fie algoritmul Ordered-Scheduling Algorithm (cursul 2, slide-ul 42), care implică algoritmul descris anterior (slide-ul 19) la care adăugăm o preprocesare cu care sortăm descrescător activitățile după timpul de desfășurare. Th. 2 afirmă că acest algoritm este $\frac{3}{2} - aproximativ$. Arătați că acest factor de aproximare poate fi îmbunătățit la $\frac{3}{2} - \frac{1}{2m}$ (unde m este numărul de calculatoare pe care se pot executa activități). (2p)

**Rezolvare**

Fie $k$ indicele masinii cu loadul maxim in urma executarii algoritmului.
$ALG = load(k)$
Fie $q$ ultima activitate adaugata pe masina $k$
Fie $load'(i)$ load-ul masinii $i$ fix inainte ca activitatea $q$ sa fie asociata masinii $k$.
$ALG = load(K) = load'(k) + t_q$
Conform justificarii de la curs:

$$load'(k) + t_q \leq \frac{1}{m} \sum_{i=1}^{n} t_i + t_q$$

Dar, faptul ca adaugam activitate $q$ la masina $k$ ne spune ca $load$‘$(k)$ este min(load‘). Astfel, putem imbunatati inegalitatea:

$$load'(k) + t_q \leq \frac{1}{m} \sum_{i=1}^{q-1} t_i + t_q$$

Daca $q \leq m$ atunci $q$ va fi pusa peste o masina goala, deci $ALG = OPT$ Daca $q > m$ atunci:

$$\begin{aligned} load'(k) + t_q &\leq \frac{1}{m} \sum_{i=1}^{q-1} t_i + t_q = \frac{1}{m} \sum_{i=1}^{q} t_i + (1 - \frac{1}{m}) t_q \\ &\leq \frac{1}{m} \sum_{i=1}^{n} t_i + (1 - \frac{1}{m}) \cdot \frac{1}{2}(t_m + t_{m+1}) \\ &\leq OPT + \frac{1}{2} OPT - \frac{1}{2m} = (\frac{3}{2} - \frac{1}{2m}) OPT \end{aligned}$$