

KNAPSACK

1. Q. // include <iostream>

using namespace std;

ifstream in ("knapsack.in");

ofstream out ("knapsack.out");

int V[105], dp[105][105];

int main()

{ int n, k;

in >> n >> k;

for (int i = 0; i < n; i++) in >> V[i];

for (int i = 1; i <= n; i++)

for (int j = 1; j <= k; j++)

{ if (V[i] <= j)

dp[i][j] = max(dp[i - 1][j], dp[i - 1][j - V[i]] + V[i]);

else

dp[i][j] = dp[i - 1][j];

out << dp[n][k];

return 0;

}

b. ~~XX~~ include <fstream>

using namespace std;

int main()

{ int k, x, suma = 0;
cin >> k;

while (cin >> x)

{ if (suma + x ≤ k)

 suma = suma + x;

 else

 suma = max(suma, x);

cout << suma;

return 0;

În timp ce rămăști fiecare următoră să
ță adunăm la suma, fără să depășim k.
Când avem o sumă mai mică de $k/2$ și
încercăm să adunăm x avem 2 cazuri:

- suma + x depășește k = $x > k/2$ și îl adunăm
- suma + x ≤ k, adunăm x și trecește mai
departe.

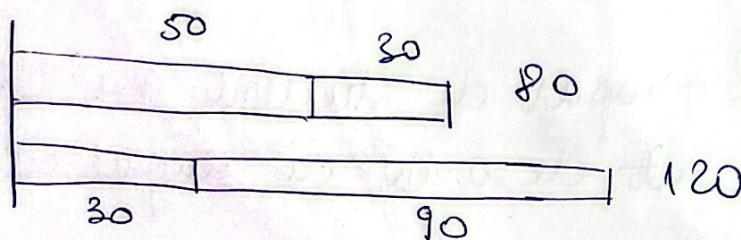
LOAD BALANCING

1.a. Este posibil ca factorul de aproximare să fie corect.

Exemplu:

Pentru setul $\{50, 30, 90, 30\}$. obt. organiza-

rea:



Organizarea este optimă ($\max(50+30, 30+90) = 120$)

Cum $120 \times 1.1 = 132 > 120 \Rightarrow OPT \leq ALG \leq 1.1$

- b. Pentru orice set de activități cu timpul de lucru și mult > 10 algoritmul Optiu așează toate activitățile în mod echilibrat
- \Rightarrow diferența maximă dintre încărățurile celor 2 mașini va fi ≤ 10 .
 - \Rightarrow În cazul optim, notăm $D_{OPT} =$ diferența maximă dintre încărățurile celor 2 mașini pt. algoritmul optim și caușul setului de activități cu timpul max. 10.
 - $\Rightarrow D_{OPT} \leq 10$

$D_{ALG} =$ diferența dintre încărățurile celor 2 mașini pt. algo. propus de student în caușul setului de act. cu timp max 10.

$$D_{ALG} = 40 \quad (\text{Jucăreatura primului matiu} = 80, \\ \text{Jucăreatura de pe a 2-a matiu} = 120)$$

Dar pt. ca algoritmul să fie 1,1-aprox,

$$D_{ALG} \leq 1,1 * D_{OPT}$$

$$\text{Dar } D_{OPT} \leq 10 \quad | \Rightarrow D_{ALG} \leq 11 \quad \times_0$$

=> Algoritmul propus de student nu este 1,1-aprox
pentru un set de activ.^{fiec.} timpul cel mult 10,

3. - $m = \text{nr. de masini}$

- $n = \text{nr. de joburi care trebuie procesate}$

- $t_j = \text{timp necesar pt. executia jobului } j (1 \leq j \leq n)$.

Din ipoteza: $t_1 \geq t_2 \geq \dots \geq t_n$

Fol. urm. leme:

Lemă 1: $\text{OPT} \geq \max\left(\frac{1}{m} \sum_{1 \leq j \leq n} t_j, \max\{t_j | 1 \leq j \leq n\}\right)$

Lemă 2: $m > m : \text{OPT} \geq t_m + t_{m+1}$

Pt. Lemă 1: $\max\{t_j | 1 \leq j \leq m\} = t_1$.

$$\Rightarrow \text{OPT} \geq \max\left(\frac{1}{m} \sum_{1 \leq j \leq n} t_j, t_1\right)$$

Aveam:

- $k = \text{indicele masinii cu load max. după exec. algoritmului}$

- $g = \text{indicele ultimului job adăugat masinii } k$

- $\text{load}'(m) = \text{load-ul masinii } m \text{ după ce au}$
 $\text{repartizat primele } g-1 \text{ joburi (nu } \neq g\text{).}$

$$\Rightarrow \text{Aveam } ALG = \text{load}'(k) + t_g.$$

\Rightarrow Aveam 2 cazuri $\begin{cases} 1. g \leq m \\ 2. g > m \end{cases}$

Caz 1 $g \leq m \Rightarrow g \text{ va fi adăugată unei masini fără incarcatura.} \Rightarrow \text{Masinii } k \text{ și } g \text{ nu vor fi adăugata activitatea } g \text{ (ca singura activitate)}$

$$\Rightarrow ALG = t_g \leq t_1$$

Dim Lemma 1 : $t_1 \leq OPT$

$$\Rightarrow ALG \leq OPT \Rightarrow ALG = OPT$$

Case 2 $2 > m$

$$load'(K) \leq \frac{1}{m} \sum_{1 \leq i \leq m} load'(i) \leq \frac{1}{m} \sum_{1 \leq j \leq 2} t_j$$

$$\frac{1}{m} \sum_{1 \leq j \leq 2} t_j \leq \frac{1}{m} \left(\sum_{1 \leq j \leq n} t_j - t_2 \right) = \frac{1}{m} \sum_{1 \leq j \leq n} t_j - \frac{1}{m} t_2$$

Dim Lemma 1 : $\frac{1}{m} \sum_{1 \leq j \leq n} t_j \leq OPT$

$$\text{Cum } load'(K) \leq \frac{1}{m} \sum_{1 \leq j \leq n} t_j - \frac{1}{m} t_2 \quad \boxed{=}$$

$$\Rightarrow load'(K) \leq OPT - \frac{1}{m} t_2$$

$$\Rightarrow \underline{ALG} = load'(K) + t_2 \leq OPT - \frac{1}{m} t_2 + t_2$$

$$\Rightarrow ALG \leq OPT + t_2 \left(1 - \frac{1}{m}\right)$$

$$2 > m \Rightarrow 2 \geq m+1$$
$$t_1 \geq t_2 \geq \dots \geq t_m$$
$$\Rightarrow t_2 \leq \frac{t_m + t_{m+1}}{2} \quad \boxed{\Rightarrow}$$

$$\Rightarrow ALG \leq OPT + \frac{1}{2} (t_m + t_{m+1}) \left(1 - \frac{1}{m}\right) \quad \boxed{\Rightarrow}$$

Si dim Lemma 2, $t_m + t_{m+1} \leq OPT$

$$\Rightarrow \text{ALG} \leq \text{OPT} + \frac{1}{2} \left(1 - \frac{1}{m}\right) \text{OPT}$$

$$\text{ALG} \leq \text{OPT} \left(1 + \frac{1}{2} - \frac{1}{2m}\right)$$

$$\text{ALG} \leq \text{OPT} \left(\frac{3}{2} - \frac{1}{2m}\right) \Rightarrow$$

Alg. este $\frac{3}{2} - \frac{1}{2m}$ aproximativ.

TSP

1. a) Problema rămâne NP-hard.

Presupunem prim absurd că problema nu rămâne NP-hard.

Pt. orice graf $G(V, E)$, putem construi graful $G'(V, \bar{E})$, în care muchiile sunt de cost 1 dacă sunt și în G și de cost 2 altfel. (G' este graf "complet").

Astfel graful G' îndeplinește condiția din enunt și putem aplica TSP pe el.

Când aplicăm TSP, obținem costul total minim = $m \Leftrightarrow$ graful inițial G este hamiltonian.

⇒ Algoritmul se reduce la ~~o~~ genial de determinare a unui ciclu hamiltonian.

Cum alg. de det. pt. ciclu hamiltonian este în NP-hard.

⇒ Algoritmul nostru în graful G' este în NP-hard.

b) Avem 4 cazuri posibile: $\{1,1,1\}$, $\{1,1,2\}$,
 $\{1,2,2\}$, $\{2,2,2\}$. Cum toate sunt
 satisfac ineq. triunghiului \Rightarrow Ponderele
 sunt satisfac. ineq. triunghiului.

c) Vom demonstra im aceasta instanta alg. din curs
 nu este $\frac{3}{2}$ -aproximativ.

ALG este $\frac{3}{2}$ -aproximativ daca: $ALG < \frac{3}{2} \cdot OPT$

Luam graful complet $H(V, E)$ in care toate
 muchile au costul 1.

In acest caz, stim ca $OPT = n \Rightarrow \sum_{e \in E} OPT = \frac{3n}{2}$.

Dar daca aplicam alg. din curs, parcurgem
 APM-ul de 2 ori si APM-ul are exact $n-1$
 muchii $\Rightarrow ALG = 2(n-1)$.

Cum $ALG = 2(n-1) > \frac{3n}{2} = \frac{3}{2} \cdot OPT \Rightarrow$

\Rightarrow Algoritmul nu este $\frac{3}{2}$ -aproximativ in acest
 caz.

VERTEX COVER

1.a) Exemplu de worst case:

$$(\#_1 \vee \#_1 \vee \#_1) \wedge (\#_1 \vee \#_1 \vee \#_2) \wedge (\#_1 \vee \#_1 \vee \#_3) \wedge \dots (\#_1 \vee \#_1 \vee \#_n)$$

În cazul optim: Alegem $\#_1$ la primul pas.

Worst Case: În primul pas alegem $\#_2$, apoi $\#_3, \dots, \#_n$
și la ultimul pas $\#_1 \Rightarrow$ Multimea variabilelor
care trebuie să fie true este $\{\#_2, \#_3, \dots, \#_n, \#_1\}$.

⇒ În worst case factorul de aproximare este n ,
cu $n =$ cardinalul lui X .

b) Algoritm 3-aproximativ pt. problema liniit:

1. $C = \{C_1, \dots, C_m\} =$ mult. de predicate
 $X = \{\#_1, \dots, \#_n\} =$ mult. de variabile

2. Cât timp $C \neq \emptyset$ executa:

2.1. Alegem aleator $C_j \in C$

2.2. $\#_i \leftarrow \text{true}, \forall \#_i \in C_j$

2.3. Eliminăm din C toate predicatele
ce îl contin pe $\#_i, \forall \#_i \in C_j$.

3. return *

În worst case, în predicat avem 3 variabile diferite,
pe care le eliminăm desigur, la un pas.

⇒ La fiecare pas eliminăm între 1 și 3 variabile.
Algoritmul optim selectată exact un element

la fiecare pas. \Rightarrow Putem considera că algoritmul propus este 3-approximativ.

c. $X = \{x_1, \dots, x_n\}$, $t_i \in \mathbb{R}$, $i = 1, \dots, n$

$$C = \{c_1, \dots, c_m\}$$

În fiecare predicat avem p variabile

ce $p \leq 3$. Avem urm. prop.:

1. (\forall) $c_i \in C$: $\sum_{i=1}^p t_i \geq 1$

2. $0 \leq t_i \leq 1$, (\forall) $i \in \{1, \dots, n\}$

Minimizati suma: $\sum_{i=1}^p t_i$.

d. Transformăm t_i în $f(t_i)$, pentru că vom să mergem ce true/false, deci trebuie să transform valori reale în boole 0/1.

De transf. după urm. regulă:

$$f(t) = \begin{cases} 0, & t < \frac{1}{3} \\ 1, & t \geq \frac{1}{3} \end{cases}$$

Fie $OPT =$ sol. optimă pt. problema

$T =$ mult. de variabile care sunt true la terminarea algoritmului.

$ALG =$ Val. rez. după rularea algoritmului.

Alg. are val. finală: $ALG = \sum_{1 \leq i \leq n} f(t_i)$.

$$f(t_i) = 1 \Leftrightarrow t_i \in T \Rightarrow ALG = |T|.$$

$$\text{Dacă } t_i \in T \Leftrightarrow t_i \geq \frac{1}{3} \Leftrightarrow 3t_i \geq 1 \Rightarrow ALG \leq \sum_{t_i \in T} 3t_i \leq 3|T|$$

Suma $t_i \in T \leq$ suma tuturor t_i .

$$\Rightarrow \text{ALG}_1 \leq 3 \sum_{x_i \in T} t_i \leq 3 \sum_{1 \leq i \leq n} t_i \leq 3 \text{OPT}.$$

Cum $\text{OPT} \leq \text{ALG}_1 \leq 3 \text{OPT}$ =

Algoritmul este 3-aproximativ.