

Exemple de vérification pour 3SAT avec comme paramètre ϵ

→ $O(\log m)$ bits aléatoires

→ 3 bits consultés ($O(1)$)

→ probabilité d'erreur $\leq 1 - \frac{1}{m}$

$m = \text{nbre de clauses de la formule 3SAT } \phi$
↳ probabilité d'accepter une formule non satisfiable

pour $P(P(\log m, 1) = NP)$ ce serait $< \frac{1}{2}$

→ Si ϕ satisfiable, la probabilité d'accepter = 1

Fonctionnement : Utiliser les $O(\log m)$ bits pour choisir une clause (parmi m) dans ϕ

- les 3 bits de la preuve à regarder = les valeurs des 3 variables apparaissant dans la clause

$$\phi = (x_1 \wedge x_2 \vee x_4) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4) \wedge (x_2 \wedge x_3 \wedge x_4)$$

↓
clause choisie aléatoirement

preuve :

$$y = \begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = 1 \\ x_4 = 1 \end{cases}$$

3 bits de la preuve à regarder.

⇒ on regarde si la clause est satisfait.
si satisfait ⇒ accepter la preuve
sinon ⇒ refuser la preuve.

• si ϕ est satisfiable ⇒ on peut trouver des valeurs aux variables tq ϕ est satisfait

⇒ pour cette preuve (= affectation des valeurs aux variables) toutes les clauses sont satisfaites ⇒ probabilité d'accepter cette preuve = 1

• si ϕ n'est pas satisfiable ⇒ probabilité d'accepter ~~des preuves~~
pour toutes les preuves $\leq 1 - 1/m = \frac{m-1}{m} = \frac{\text{max clauses satis}}{\text{nbre clauses}}$

Problème du sac-à-dos.

(2)
22/21

- m objets
 - $v_i, p_i = P_i$ profit de l'objet i
 - v_i = volume objet i
 - C = capacité du sac
 - sous ensemble d'objets tq la capacité n'est pas dépassée et le profit est maximum
- $\max \sum_{i \in S} p_i, \text{ st: } \sum_{i \in S} v_i \leq C.$

Algorithme programmation dynamique:

$G(i, K)$ = gain possible maximum d'un sous ensemble des objets de $1 \rightarrow i$ avec comme capacité K

$$G(i, K) = \text{Max} \left\{ G(i-1, K); \text{ on ne met pas l'objet } i \text{ dans le sac} \right\}$$

$$G(0, K) = 0$$

$$\begin{aligned} & G(i-1, K-v_i) + p_i \\ & \text{ si } v_i \leq K \end{aligned}$$

$$G(i, 0) = 0$$

Tableau : 2D : $|m = \text{nbre d'objet}$
 $|C = \text{capacité du sac}$
 $\Rightarrow O(mC)$

Autre version : pour le profit P on s'intéresse au calcul ~~de la toute~~ ~~min~~ min nécessaire du volume permettant d'obtenir ce profit.

$A(i+1, P) = \text{volume min nécessaire pour obtenir le profit } = P \text{ sur les objets de } 1 \rightarrow i$
 $= +\infty \text{ si } P \text{ n'est pas atteignable}$

$$A(i+1, p) = \min \{ A(i, p); \quad (i+1 \text{ n'est pas dans le sac}) \quad ③ \\ A(i, p - p_{i+1}) + v_{i+1} \} \quad (i+1 \text{ dans le sac}) \\ \text{ si } p_{i+1} \leq P$$

on trouve la solution en prenant le volume $\leq C$
qui donne le meilleur profit
 $\max \{ p \mid A(m, p) \leq C \}$

borne sur le profit atteignable : $P_{\max} = \max_{i=1 \dots m} p_i$
 $\rightarrow \sum_{i=1}^m p_i \leq m P_{\max}$
 \Rightarrow complexité $O(m \times m \times P_{\max}) = O(m^2 P_{\max})$

algo pseudo polynomial \rightarrow polynomial en m et P_{\max}
- $m \rightarrow$ taille de la donnée : on a donné 2^n valeurs
- $P_{\max} \rightarrow$ valeur d'un des p_i
 \rightarrow taille en binaire : $\log_2 P_{\max}$
 \Rightarrow exponentiel en ~~par~~ la taille de P_{\max}

Définition un problème T_i est pseudo polynomial s'il peut être résolu par un algorithme qui s'exécute sur toute instance x en temps borné par un polynôme en $|x|$ et en $\max(x)$ ~~ou~~ $\max(x)$ et la taille de la plus grande valeur donné dans x .

Schéma d'approximation polynomial : PTAS
soit T un problème d'optimisation NP-dur

$\rightarrow f_T =$ fct objectif

A est un schéma d'approximation pour T
avec la donnée (x, ϵ) où $x =$ instance de T
et $\epsilon > 0$ un paramètre d'erreur, si il produit une solution s telle que

$$- f_T(x, s) \leq (1+\epsilon) \text{OPT} \quad (T=\min)$$

$$- f_T(x, s) \geq (1-\epsilon) \text{OPT} \quad (T=\max)$$

soit $\epsilon = 10^{-6}$ l'objectif pour la ST. S sur l'entrée

et c'est un schéma d'approx polynomial, si $\epsilon > 0$
A s'exécute en temps polynomial en $|x| = \text{taille de l'inst}$

Remarque: si on choisit $\epsilon > 0$ très petit on peut s'approcher
aussi près qu'on veut de la solution optimale
- $A(x, \epsilon)$ polynomial en $|x|$ mais dépend
aussi de $\epsilon \rightarrow$ peut être exponentiel en $\frac{1}{\epsilon}$
ou dépendre de $1/\epsilon$ avec n'importe quelle fonction
avec une croissance très rapide.

\Rightarrow on peut s'approcher autant qu'on veut de la
valeur optimale, mais ça peut prendre très très
longtemps.

\hookrightarrow PTAS = polynomial Time Approximation Scheme

FPTAS : Fully polynomial Time Approximation Scheme
 \rightarrow et doit être polynomial en $|x|$ et en $\frac{1}{\epsilon}$

\Rightarrow beaucoup plus rassurant pour le temps de calcul

Mais: ça n'existe que pour une petite partie des
problèmes d'optimisation.

FPTAS pour le sac à Dos

- l'algorithme de programmation dynamique est de complexité
 $O(n^2 P_{max}) \rightarrow$ pseudo polynomial à cause de P_{max}

\rightarrow essayer de réduire artificiellement P_{max}

1. soit $K = \frac{\epsilon P_{max}}{m} \rightarrow$ diviser les profits par K
pour $\epsilon > 0$

2. on définit les profits $\tilde{p}_i = \left\lfloor \frac{p_i}{K} \right\rfloor$

3. utiliser l'algorithme de programmation dynamique pour
obtenir une solution S''

4. $S' =$ solution trouvé par l'algo (FPTAS)

1] L'ensemble S' retourné par l'algorithme précédent vérifie

$$\sum_{i \in S'} p_i = \text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT} = (1-\varepsilon) \times \text{valeur optimale}$$

preuve: soit Θ une solution optimale pour le problème de départ (avec p_i et pas p'_i)

- $\forall i \in 1..m$, $p'_i = \lfloor \frac{p_i}{K} \rfloor$ donc $Kp'_i \leq p_i$
et $p_i - K \leq Kp'_i \leq p_i$

$$\frac{p_i}{K} - 1 \leq p'_i = \left\lfloor \frac{p_i}{K} \right\rfloor \leq \frac{p'_i}{K}$$

$$\Rightarrow p_i - K \leq Kp'_i \leq p_i \quad \text{①} \quad \text{②}$$

- on somme sur tous les objets $\in \Theta$

$$\cancel{\text{OPT}} = \sum_{i \in \Theta} p_i \leq K$$

$$\cancel{\text{OPT}} = \sum_{i \in \Theta} p_i = 10|K| \stackrel{\text{①}}{\leq} K \sum_{i \in \Theta} p'_i \leq \sum_{i \in \Theta} p_i \stackrel{\text{OPT}}{\leq}$$

$$\sum_{i \in \Theta} p_i - K \sum_{i \in \Theta} p'_i \leq K|10| \leq km$$

car Θ contient au plus m éléments

- S' solution de l'algo ~~d'acc~~ de programmation dy namique
 $\Rightarrow S'$ est optimal pour le problème avec les p'_i

$\sum_{i \in S'} p'_i \geq \sum_{i \in \Theta} p'_i$ car pour ce problème avec les p'_i , S' est optimale et Θ est a priori seulement réalisable $\Rightarrow S'$ meilleur Θ .

$$\sum_{i \in S'} p_i \stackrel{\text{②}}{\geq} k \sum_{i \in S'} p'_i \stackrel{\text{③}}{\geq} K \sum_{i \in \Theta} p'_i \geq \sum_{i \in \Theta} p_i - Km$$

$$\begin{aligned}
 \text{Valeur de } S' &= \sum_{i \in S'} p_i \geq \sum_{i \in S} p_i - K_m = OPT - \frac{\epsilon P_{\max}}{gt} \times gt \\
 &= OPT - \epsilon P_{\max} \\
 &\geq OPT - \epsilon OPT = (1-\epsilon)OPT
 \end{aligned} \tag{6}$$

$OPT \geq P_{\max}$: on doit pouvoir choisir uniquement l'objet tq $p_i = P_{\max}$

2) l'algorithme est bien FPTAS.

Algo de programmation dynamique en

$$O(n^2 \left\lfloor \frac{P_{\max}}{K} \right\rfloor) = O(n^2 \left\lfloor \frac{n}{\epsilon} \right\rfloor)$$

$$\frac{P_{\max}}{K} = \frac{P_{\max} \times n}{\epsilon \times P_{\max}} = \frac{n}{\epsilon}$$

\Rightarrow polynomial en n et $\frac{1}{\epsilon} \Rightarrow$ FPTAS.

Exo Subset-Sum ratio problem

- m entiers ≥ 0 $a_1 < a_2 \dots < a_m$

- Trouver 2 sous ensembles $S_1, S_2 \subseteq \{1, \dots, m\}$

avec $\sum_{i \in S_1} a_i \geq \sum_{i \in S_2} a_i$ tq

$$\frac{\sum_{i \in S_1} a_i}{\sum_{i \in S_2} a_i} \text{ est minimum}$$

\rightarrow Trouver un ~~algo~~ schema FPTAS pour ce problème (passer par algo pseudo-polynomial et technique similaire à sac-à-dos)
(prouver!)