# Knapsack

Имаме раница с обем $V$ и
предмети като всеки предмет $i$
има размер $s_i$ и стойност $v_i$
size, volume, value
$(s_i \in \mathbb{Z}^+$, а $v_i \in \mathbb{R}^+)$.

Целта ни е да изберем такова подмножество на предметите, т.е. да не надвишим обема на раницата и в същото време да максимизираме ст-та на всички предмети в нея.

Вариант 1: Имаме неограничен брой от всеки предмет. (позв. неогран. повторения)

Вариант 2: Имаме краен брой от всеки предмет. (може този краен брой като отделни различни предмети да си ги запишем, т.е без неограничени повторения).

Ако имаме ограничен брой предмети от всеки тип, то нека да ги гледаме все едно са си отделни типове сами по себе си.

① # подпроблеми: във всеки момент

ще сме запълнили определен обем от раницата $U$ и ни остава свободно $V-U$ и ще се опитваме да видим дали от префикса $A[1,..,i]$ можем да изберем такава подредица от предмети т.е. да е максимално value и да не надвишава $V-U$. Т.е имаме $\Theta(n.V)$ ← обем на раницата.
брой типа предмети

② Избор (решаване на подпроблем).
Имаме капацитет $V' \le V$ и избираме от префикс $A[1,..,i]$. За всеки предмет $A[i]$ имаме избор да го добавим, ако не надвишава $V'$ или просто като цяло да го пропуснем даже и $s_i \le V'$.
Значи във всеки подзадача извършваме работа отнемаща константно време за решаването й.
Тогава: $\Theta(1)$.

③ Рекурентно уравнение:

Тогава имаме таблица $dp[0 \dots n][0 \dots V]$ като base case са:

$$dp[0][0] = 0$$

$$dp[i][0] = 0 \quad 1 \le i \le n$$

$$dp[0][i] = 0 \quad 1 \le i \le V$$

Оттам нататък:

$$dp[i][j] = \max\{ \overbrace{dp[i-1][j-s_i] + v_i}^{\text{include}} \ (\text{if } s_i \le j), \ \underbrace{dp[i-1][j]}_{\text{skip}} \}$$

for $i = 1, \dots n$; for $j = 1, \dots V$.

Т.е. $dp[i][j]$ = макс. ст-т на раницата с обем $j$ и избор измежду предметите $A[1, \dots, i]$.

④ Виждаме и топ. наредбата: от по-малко раница и по-къси подредици от предмети към по-големи.

⑤ Решението на оригиналната задача е в $dp[n][V]$.

HW Помислете си вариант 1 на задачата

Сложности: $T(n,V) \simeq \Theta(n.V)$ и $M(n,V) \simeq \Theta(n.V)$ като това е псевдополиномиално, т.к. зависи от размера $V$ (за колко бита ни трябват да го запишем: $\log V$).

Приложение

1 От ДИ

Имате $n>0$ книжки като във всяка от тях имате $a_i > 0$ страници.
Търсим $I = \{i_1, \dots, i_k\} \subseteq \{1, \dots, n\}$ т.ч.
$\sum_{i \in I} a_i \to max$ и да се дели на 3

---

Алчна стратегия ще бъде по-трудна за измисляне тук.
Затова нека да забележим, че има нещо общо с knapsack.
Нас ни интересува макс $\sum_{i \in I} a_i$ и $3 \mid \sum_{i \in I} a_i$, но да забележим, че във

Всеки един момент ние разглеждаме някакъв префикс на масива $A[1,..,n]$ и се питаме коя щяла коя е подредицата, т.е. да е максимална сумата и (релаксиране на условието) дава остатък $j \in \{0,1,2\}$ по модул 3.

Много важно да включим в разсъжденията ни оптималните решения даващи различен остатък от желания $\underline{0}$, защото може в последствие оптималното решение на основната задача да зависи от някои от тях!

Т.е. имаме тогава следното:

$$dp[1][A[1]\%3] = A[1]$$

$$dp[1][k] = 0, \text{ за } k \in \{0,1,2\} \setminus \{A[1]\%3\}$$

$$dp[i][j] = \max\{\underbrace{dp[i-1][j]}_{\text{skip } A[i]},\ dp[i-1][((\underbrace{j - A[i]\%3}_{\text{може да е отрицателно}}) + 3)\%3] + A[i]\}$$

for $i = 2,..,n$

for $j = 0,1,2$

Нашето решение е в dp[n][0].

$T(n) \asymp \Theta(n.3) = \Theta(n)$

(2) Вижте в pdf, който съм

(HW) качил задачите:

- DynProgExam.pdf.

## Longest common subsequence

Дадени са думи $u$ и $v$. Намерете най-дългата им последователност обща подредица от букви. Нека $|u|=n$, $|v|=m$.

Прим: springtime    lcs: printi
printing

Brute force:

```
lcs(u, v, i, j)
  if i ≤ 0 ∨ j ≤ 0
    return 0
  else if u[i] = v[j]
    return 1 + lcs(u, v, i-1, j-1)
  else
    return max(lcs(u, v, i-1, j),
               lcs(u, v, i, j-1))
```

1. #подпроблеми: $\Theta(|u|.|v|)$
2. Guess: $u[i]=v[j]$ $\begin{cases} \text{true} \to 1 + \text{опт. реш за } \langle i-1, j-1\rangle \\ \text{false} \to \text{максимум от оптималните решения за } \langle i-1, j\rangle \text{ и } \langle i, j-1\rangle \end{cases}$
   $\Theta(1)$

③ $dp[0][0] = 0$

$dp[i][0] = 0, \quad 1 \le i \le n$

$dp[0][j] = 0, \quad 1 \le j \le m$

$$dp[i][j] = \begin{cases} 1 + dp[i-1][j-1], & u[i] = v[j] \\ \max(dp[i-1][j],\ dp[i][j-1]) & \text{else.} \end{cases}$$

for $i = 1, \dots, n$
for $j = 1, \dots, n$

④ От къси към дълги префикси.

⑤ Решение в $dp[|u|][|v|]$

$T(n,m) \asymp \Theta(n,m)$ и $M(n,m) \asymp \Theta(n,m)$.

# The coin change problem

Работите на каса в супермаркет и имате различни типове стотинки $n$ и то в безкрайни количества. Стойността на всяка монета все я знаете. Как можете да определите по колко начина можете да върнете ресто с тези стотинки при дадено $S$, което ви подава клиента.

Прим: имате 4 типа стотинки: 8, 3, 1, 2 и съответно за $S=3$ → имаме три начина: $\{1,1,1\}$, $\{1,2\}$ и $\{3\}$.

Т.е. търсим броят на тези Андгулации-та, i.e.: $\underset{\text{multiset}}{A \subseteq} 1 \cup 2 \cup 3 \cup 8$ :

$$\Sigma A = S$$

Подобно на SubsetSum нали? Но там търсихме съществуване тук начини!

Така трябва ни пак $dp[0..n][0..S]$

и $dp[0][0]=1$ // като вземем $\emptyset$ правим сума 0

$dp[0][i]=0$, $1 \le i \le S$

$dp[i][0]=1$, $1 \le i \le n$ // пак като вземем $\emptyset$ сме окей

$dp[i][j]=0$, $1 \le i \le n$ & $1 \le j \le S$

for $i = 1, \dots, n$
  for $j = 1, \dots, S$
    $dp[i][j] \mathrel{+}= \underbrace{dp[i-1][j]}_{\text{exclude } A[i]} + \underbrace{dp[i][j-A[i]]}_{\text{include } A[i]}$

Отговорът ни е в $dp[n][S]$.

(HW) Помислете как ще стане, ако имаме краен брой монети.
Ако искаме пак да намерим минимално по мощност представяне на $S$ като сума от стотинките зададени?

# minStops

Трябва да пътуваме от град A до град B и разстоянието м/у тях е dist. По пътя има бензиностанции на разстояния $d_1 < d_2 < .. < d_n$ от A. С пълен резервоар колата изминава k километра и е пълен в началото на пътуването.
Предложете алгоритъм, т.е. за време O(n) съставя възможно най-кратък списък от бензиностанции, на които да спрете да зареждате.

С алчен алгоритъм се постига решение с такава сложност.
Алчните алгоритми имат такова име, т.к. избират най-доброто предложение в момента без да мислят за напред. Такъв алгоритъм въобще не би успял да играе шах, но при някои примери вероятно ще е добър. Примери

за такива алгоритми са Prim, Kruskal, Huffman, Set cover, Interval scheduling и сега и тази задача.

Сега стратегията ни ще е следната: ще спираме възможно най-късно т.е. чак на последната бензиностанция преди да ни свърши бензина. Така:

```
minStop(d[1,..,n], dist, k)
stops[1,..,n], idx ← 1
lastStop ← 0
for i ← 2 to n do:
    if d[i] - lastStop > k
        stops[idx] ← d[i-1]
        idx ← idx + 1
        lastStop ← d[i-1]
if dist - lastStop > k
    stops[idx] ← d[n]
return <stops, idx>.
```