# Динамическое программирование

- Кратч. путь в ор. ацикл. графе
- НВП $O(n^2)$

---

## Задача о рюкзаке

Рюкзак объёмом $W$, $n$ вещей

$c_i$ – цена $i$-й вещи
$w_i$ – объём $i$-й вещи, $>0$

$i_1, i_2, \dots, i_k$ $\begin{cases} \sum_{j=1}^{k} w_{i_j} \le W \\ \sum_{j=1}^{k} c_{i_j} \to \max \end{cases}$

Наивное решение: $O(2^n)$

① Каждый предмет можно брать неогр. кол.-во раз

1). $d[w]$ = max сумма стоимостей предметов, сумма объёмов которых равна $w$

2). Ответ: $\max\limits_{0 \le w \le W} d[w]$

3). Переход: $d[w] = \max\limits_{\substack{1 \le i \le n \\ w_i \le w}} (d[w-w_i]+c_i)$, $w>0$


Time $=O(nW)$
Mem $=O(W)$

(Альтернативно: переходы вперёд
$\forall w$ $d[w]$ – посчитано
for $i=1..n$
$d[w+w_i]=\max(d[w]+c_i, d[w+w_i])$)


4). Порядок обхода: $w \nearrow$
for $w=0 \dots W$ $d[w]= \dots$

5). $d[0]=0$.

② Каждый предмет берём $\le 1$ раза

1). $d[w][i]$ = max сумма стоим. предметов, если можно брать предметы $1..i$, и их суммарный объём $=w$

Time $=O(nW)$
Mem $=O(nW)$

2). $\max\limits_{0 \le w \le W} d[w][n]$ — ответ

3). $d[w][i] = \max \begin{bmatrix} d[w][i-1] \\ d[w-w_i][i-1]+c_i \end{bmatrix}$, если $w \geqslant w_i$

4). for $w=0..W$
for $i=0..n$
$d[w][i]= \dots$

5). $w=0 \Rightarrow d[w][i]=0$ $\forall i=0..n$
$i=0 \to d[0][0]=0$
$\to d[w][0]=-\infty$, $w>0$

---

Д.П. → „индукция"
→ „рекурсия с запоминанием" – только нужные состояния

пусть в задаче о рюкзаке* все веса предм. и $W$ – все делятся на 1000
рекурсия сделает в 1000 раз меньше действ.

рюкзак*: набрать ровно $W$
$d[W][n]$
$f(W,n)$

(Knapsack Problem)

---

## Редакционное расстояние (Edit Distance)

(расст. Левенштейна)

1 определение:
- Операция „вставка буквы"
- Операция „удаление буквы"
- операция „изменение буквы"

$d(s,t)$ = min операций $s \to t$

2 определение:

$-s_1-s_2---s_3\, s_n-$
$t_1 t_2 t_3 -- t_4 t_5 \dots t_m$

выравнивание

$s=s_1 \dots s_n$
$t=t_1 \dots t_m$

$\binom{-}{t_i} \to +1$

$\binom{s_i}{t_j}$ → $0$, $s_i=t_j$; → $1$, $s_i \ne t_j$

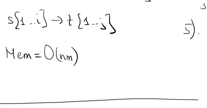
$\binom{s_i}{-} \to +1$

1). $f[i][j] = d(s[1..i], t[1..j])$

2). Ответ: $f[n][m]$

3). $f[i][j] = \min \begin{cases} f[i-1][j]+1 \\ f[i][j-1]+1 \\ f[i-1][j-1]+\begin{cases} 1, s_i \ne t_j \\ 0, s_i=t_j \end{cases} \end{cases}$ $i,j>0$

$s \to s' \to t' \to t$
удален. замены добавл.




$f[i][j]$ = min # оп., чтобы $s[1..i] \to t[1..j]$

4) $i=0$ $f[0][j]=j$
$j=0$ $f[i][0]=i$

5). for $i=0..n$
for $j=0..m$
$f[i][j]$

Time $=O(nm)$
$O(|s| \cdot |t|)$

Mem $=O(nm)$

---

## Задача Коммивояжёра

- неор. граф
- взвеш.: $\forall$ ребро имеет вес

Хотим выйти из города №1, пройти по всем городам, не заходя в один и тот же город дважды, и вернуться в город №1
min время в пути =?

Наивное решение: $\min\limits_{\sigma \in S_{n-1}} ans(\sigma)$ $O(n!)$

$w=+\infty$

Будем считать, что граф полный ($w=+\infty$ для отс. рёбер)

1). $d[v][S]$ = min путь комм. $1 \to v$, кот. прох. в точности по городам $S$

$v$ – текущий город, $1 \le v \le n$
$S$ – множество посещ. городов, $S \subseteq \{1,2,\dots,n\}$
$(S \ni v)$
$S \ni 1$

2). $1 \in S$
$v \in S$
$v \ne 1$

$d[v][S] = \min\limits_{u \in S}(d[u][S \setminus \{v\}]+w_{uv})$


3). $d[1][\{1\}]=0$ $d[1][S]=+\infty$, $S \ne \{1\}$
$d[v][S]=+\infty$, $\begin{cases} S \not\ni v \\ S \not\ni 1 \end{cases}$

Time $=O(2^n \cdot n^2)$
Mem $=O(2^n \cdot n)$

4). for $S \subseteq \{1,\dots,n\}$ $|S| \nearrow$ $1 \in S$
for $v=1..n$ $v \in S$
$d[v][S] = \min\limits_{u \in S}( \dots )$

5). $\min\limits_{1 \le u \le n}(d[u][\{1,2,3,\dots,n\}]+w_{u1})$ – ответ

---

Реализация $S \to$ храним как число

$10_{10} = 0000\,1010_2$
$\uparrow\uparrow\uparrow\uparrow\uparrow$
$4\,3\,2\,1\,0$

маска $a[i]$ → true, → false, $0 \le i < n$
$(m \gg i) \& 1$

$a[i]$=true
$m = m \,|\, (1 \ll i)$

$a[i]$=false
$m = m \,\&\, (\sim(1 \ll i))$

$m$ $\&$ $\sim(1 \ll i)$
1111 0111

$7_{10} = 111_2$
$2\,1\,0$

$(7 \gg 1) \& 1$

$2^i$

S: 0, 1, 2, 3, 4, 5, 6, …
011 100
$|S|=2$ $|S|=1$

$\boxed{S-(1 \ll v) < S}$

```
for S = 0 ... 2^n - 1   · if (S>>0)&1==1
  for v = 0..n-1          if (S>>v)&1==1
    for u = 0..n-1
      if (S>>u)&1 == 1:
        d[S][v] = min(d[S][v],
                      d[S-(1<<v)][u]+w_uv)
```