Dynamické programovanie—zhrnutie

1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.** Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **Bázové podproblémy.** Ktoré políčka nemožno vypočítať pomocou vzťahov z predchádzajúceho kroku? Aké hodnoty by mali obsahovať?
4. **Vyberieme poradie vypĺňania.** V akom poradí musíme maticu vypĺňať tak, aby sme v každom kroku mali vypočítané všetky políčka, ktoré potrebujeme na výpočet daného políčka?

Výpočtová geometria

- časť informatiky, ktorá sa zaoberá riešením geometrických problémov
- aplikácie: počítačová grafika, robotika, navrhovanie obvodov, a iné
- stará disciplína: pamätáte si na úlohy, ktoré bolo treba riešiť pravítkom a kružidlom?
  Emile Lemoine (1902) študoval, koľko operácií je potrebných na riešenie konkrétnych problémov

Najkratšia triangulácia

Úloha: Daný je **konvexný mnohouholník** s vrcholmi vymenovanými v poradí v smere hodinových ručičiek $(v_1, v_2, \ldots, v_n)$.
Nájdite **trianguláciu** s najkratšou dĺžkou.

- **konvexný mnohouholník:** ak spojíme ľubovoľné dva body na hranici, spojnica prechádza vnútrom
- **chorda:** spojnica dvoch nesusediacich vrcholov mnohouholníka
- **triangulácia:** rozdelenie konvexného mnohouholníka na neprekrývajúce sa trojuholníky pomocou nepretínajúcich sa chord
- **dĺžka triangulácie:** dĺžka použitých chord + obvod mnohouholníka

Riešenie dynamickým programovaním

Podproblém: $t[u_1, \ldots, u_\ell]$ - dĺžka najkratšej triangulácie pre mnohouholník $(u_1, \ldots, u_\ell)$
(postupnosť $u_1, \ldots, u_\ell$ vyberáme z $v_1, \ldots, v_n$ v pôvodnom poradí)

Rekurencia: Hrana (u_ℓ, u_1) je súčasťou nejakého trojuholníka v optimálnej triangulácii; nech tretí vrchol tohto trojuholníka je u_m

Cena takejto triangulácie:

$d(u_1, u_\ell) + t[u_1, u_2, \ldots, u_m] + t[u_m, u_{m+1}, \ldots, u_\ell]$

Vyskúšaním všetkých možností pre vrchol u_m dostaneme:

$t[u_1, \ldots, u_\ell] =$

$\min\limits_{1<m<\ell} \{d(u_1, u_\ell) + t[u_1, u_2, \ldots, u_m] + t[u_m, u_{m+1}, \ldots, u_\ell]\}$

Podproblém: $t[u_1, \ldots, u_\ell]$ - dĺžka najkratšej triangulácie pre mnohouholník $(u_1, \ldots, u_\ell)$
(postupnosť $u_1, \ldots, u_\ell$ vyberáme z $v_1, \ldots, v_n$ v pôvodnom poradí)

Rekurencia: $t[u_1, \ldots, u_\ell] =$
$\min_{1<m<\ell} \{d(u_1, u_\ell) + t[u_1, u_2, \ldots, u_m] + t[u_m, u_{m+1}, \ldots, u_\ell]\}$

Základné prípady: "degenerovaný" mnohouholník z dvoch vrcholov:
$t[a, b] = d(a, b)$

Postup vypĺňania: Všetky podpostupnosti pôvodnej množiny vrcholov
Zoradené podľa počtu použitých vrcholov

Časová zložitosť: $O(2^n \cdot n)$

Optimalizácia

Rekurencia: $t[u_1, \ldots, u_\ell] =$

$$\min_{1<m<\ell} \{d(u_1, u_\ell) + t[u_1, u_2, \ldots, u_m] + t[u_m, u_{m+1}, \ldots, u_\ell]\}$$

Ak máme podproblém, ktorý zodpovedá súvislej podpostupnosti pôvodných vrcholov, potom na jeho výpočet použijeme **výhradne** podproblémy, ktoré zodpovedajú súvislej podpostupnosti pôvodných vrcholov $\Rightarrow$ stačí nám počítať podproblémy pre **súvislé podpostupnosti pôvodných vrcholov**

Finálne dynamické programovanie

Podproblém: $t[i,j]$ - dĺžka najkratšej triangulácie pre mnohouholník $(v_i, \ldots, v_j)$

Rekurencia: $t[i,j] = \min_{i<m<j} \{d(v_j, v_i) + t[i,m] + t[m,j]\}$

Základné prípady: "degenerovaný" mnohouholník z dvoch vrcholov: $t[a, a+1] = d(v_i, v_{i+1})$

Postup vypĺňania: V poradí "po uhlopriečkach", t.j. najprv $t[i,j]$ také, že $j - i = 2$
potom $t[i,j]$ také, že $j - i = 3$
potom $t[i,j]$ také, že $j - i = 4$
...

Časová zložitosť: $O(n^2 \cdot n) = O(n^3)$

Najkratšia triangulácia

```
// base case - j=i+1
for i:=1 to n-1 do
  T[i,i+1]:=D[i,i+1];

for delta:=2 to n-1 do
  // cases where j-i=delta
  for i:=1 to n-delta do
    j:=i+delta; T[i,j]:=infinity;
    // try all possible triangles v_i,v_j,v_m
    for m:=i+1 to j-1 do
      cost:=D[i,j]+T[i,m]+T[m,j];
      if cost<T[i,j] then
        T[i,j]:=cost;

return T[1,n];
```

Najkratšia triangulácia: s vypísaním riešenia

```
// base case - j=i+1
for i:=1 to n-1 do
  T[i,i+1]:=D[i,i+1];

for delta:=2 to n-1 do
  // cases where j-i=delta
  for i:=1 to n-delta do
    j:=i+delta; T[i,j]:=infinity;
    // try all possible triangles v_i,v_j,v_m
    for m:=i+1 to j-1 do
      cost:=D[i,j]+T[i,m]+T[m,j];
      if cost<T[i,j] then
*       T[i,j]:=cost; M[i,j]:=m;

return T[1,n];
```

Najkratšia triangulácia: s vypísaním riešenia

```
function give_solution(i,j)
  output edge (i,j);
  if j>i+1 then
    give_solution(i,M[i,j]);
    give_solution(M[i,j],j);
```

Zhrnutie

- Technika dynamického programovania: riešenie problémov pomocou rozkladu na dobre definované podproblémy a "vypĺňanie matice"
- Príklad lodičiek s utečencami: jednorozmerná matica
- Problém batohu: dvojrozmerná matica
- Problém rozmieňania peňazí: niekedy sa dá použiť greedy algoritmus, niekedy treba dynamické programovanie
- Minimálna triangulácia: zložitejšie dynamické programovanie
  vypĺňanie matice po uhlopriečkach
  riešenie sa vypisuje rekurzívne (vždy skladáme väčší problém z dvoch podproblémov)