Opakovanie: Greedy algoritmy

- v každom kroku vezmeme lokálne optimálny krok
- ľahké na implementáciu
- obvykle veľmi efektívne (časová zložitosť)
- metóda sa často nedá použiť
- hlavný problém: dokázať správnosť

Príklady použitia:

- Výber aktivít $\Theta(n \log n)$
- Najlepšie Huffmanovo kódovanie $\Theta(n \log n)$
- Rozmieňanie peňazí (niektoré systémy) $\Theta(m)$

Opakovanie: Dynamické programovanie

- rozkladáme problém na podproblémy
- počítame optimálne riešenia pomocou rekurencií vo veľkej matici podproblémov
- ľahké na implementáciu
- hlavný problém: vymyslieť správny podproblém

Príklady použitia:

- Rozmieňanie peňazí (všeobecné) $\Theta(mS)$
- Celočíselný problém batohu $\Theta(nW)$
- Najdlhšia spoločná podpostupnosť $\Theta(mn)$
- Najkratšia triangulácia $\Theta(n^3)$

Opakovanie: Rozdeľuj a panuj

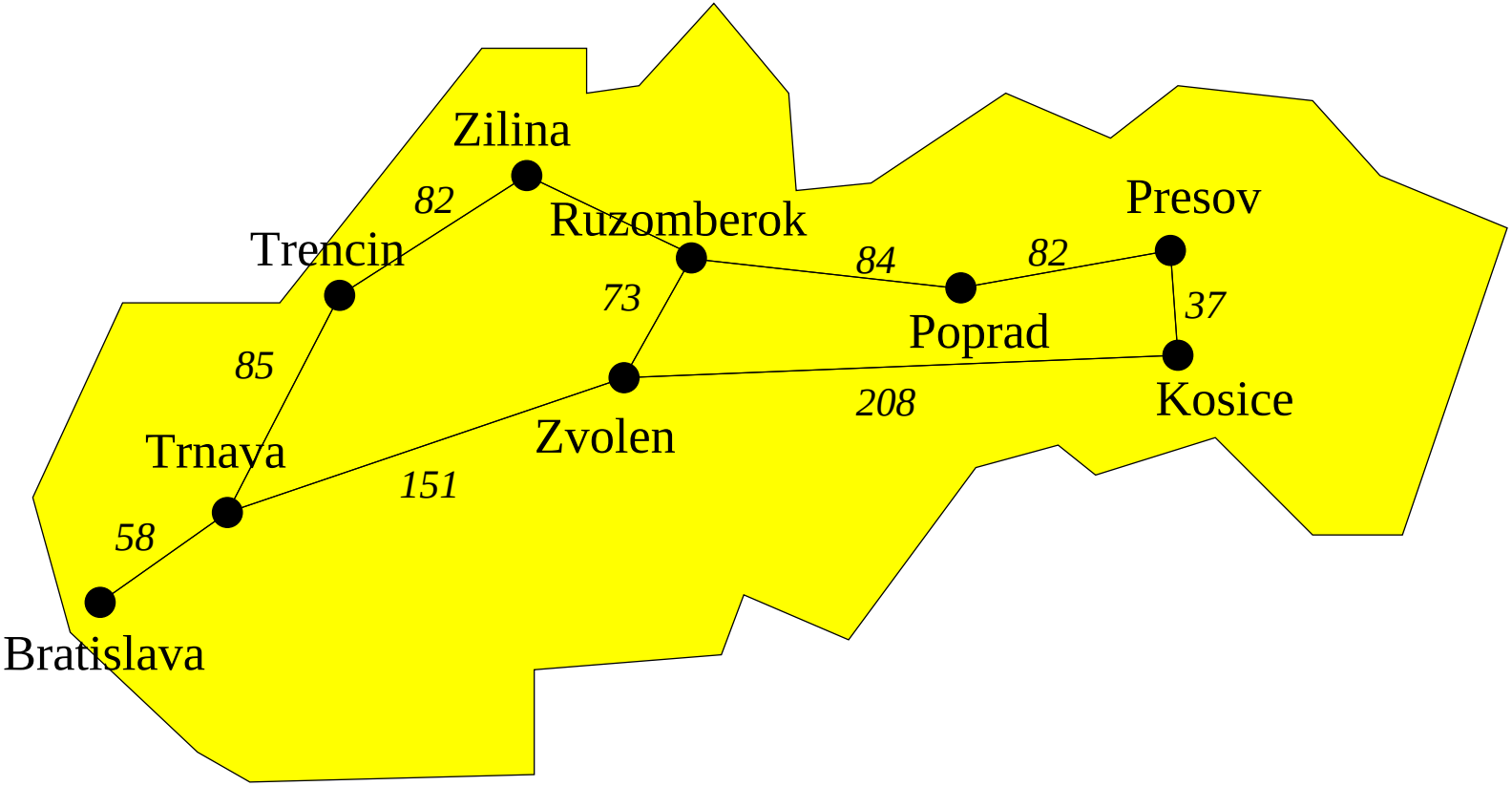
- rozdeľ problém na menšie podproblémy, vyrieš rekurzívne a skombinuj čiastkové riešenia
- niekedy ťažké na implementáciu, veľký overhead na rekurziu
- hlavný problém: analýza časovej zložitosti

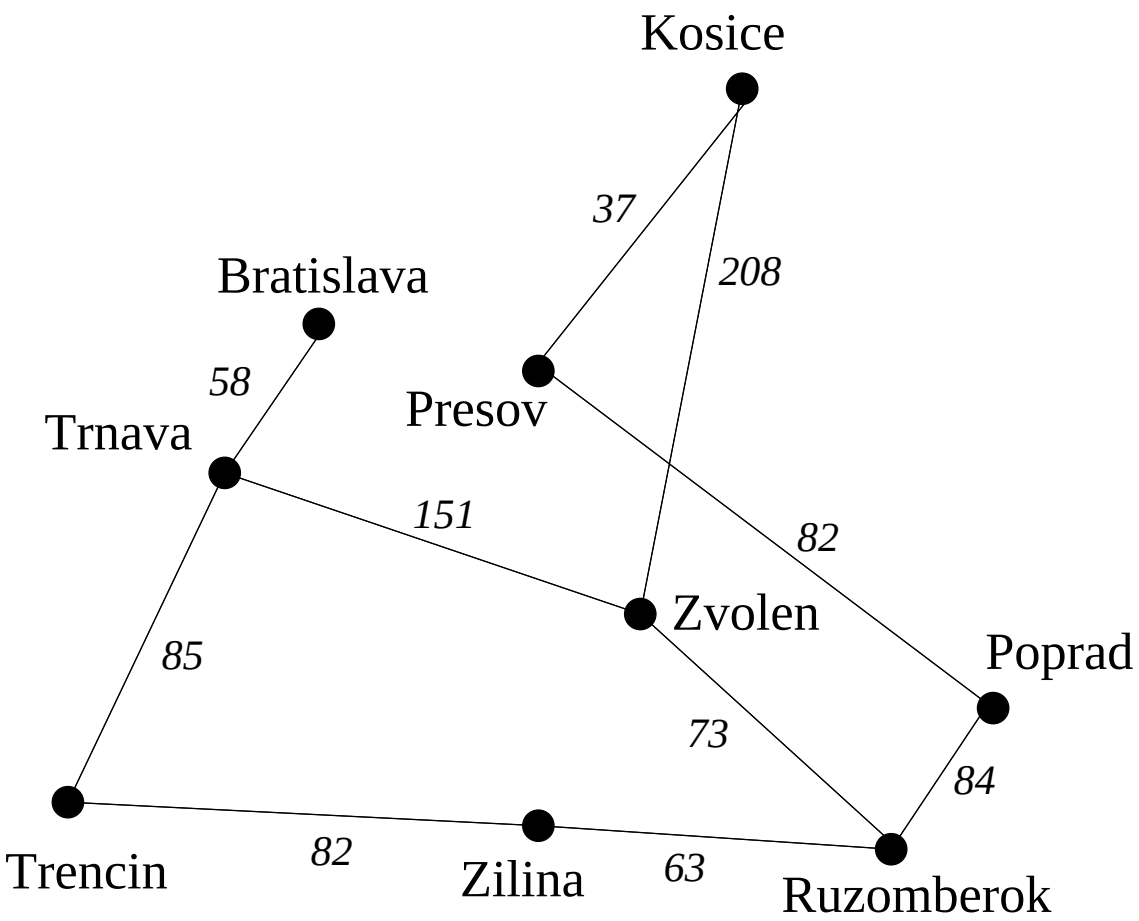
Príklady použitia:

- Triedenie (merge sort, quick sort) $\Theta(n \log n)$
- Násobenie veľkých čísel $\Theta(n^{1.58\ldots})$
- Najbližší pár bodov $\Theta(n \log n)$

Efektívne algoritmy na grafoch

- Vrcholy V ($|V| = n$), hrany E ($|E| = m$)
- Hrany môžu byť **orientované** alebo **neorientované**
- Váhované grafy: $w : E \to \mathbb{R}$
- Reprezentácia pomocou **matice susedností:**
  rýchle operácie: sú dva vrcholy spojené hranou? $\Theta(1)$
  pomalšie operácie: susedia daného vrcholu $\Theta(n)$
- Reprezentácia pomocou **zoznamov susedov:**
  všetky operácie závisia **od stupňa vrcholu**


Zilina
82
Trencin
Ruzomberok
84
82
Presov
73
37
Poprad
85
208
Kosice
Trnava
Zvolen
151
58
Bratislava


Kosice
37
208
Bratislava
58
Presov
Trnava
151
82
Zvolen
85
Poprad
73
84
82
63
Trencin
Zilina
Ruzomberok

Hľadanie najkratších ciest

Úloha: Daný je ohodnotený graf (orientovaný alebo neorientovaný)
Nájdite najkratšiu cestu z vrcholu u do vrcholu v

Dijkstrov algoritmus v skutočnosti počíta naraz najkratšie cesty z vrcholu u do **všetkých ostatných vrcholov**
váhy $w(u, v)$ **nesmú byť negatívne**

Množina dokončených vrcholov S: vrcholy u ktorých už poznáme najkratšiu cestu
Množina nedokončených vrcholov T: všetky ostatné vrcholy

Na začiatku: $S = \emptyset$, $T = V$
$dist(s)$: dĺžka najkratšej cesty z u do s, ktorá môže viesť **len cez vrcholy z S**

Dijkstrov algoritmus

```
// initialize dist, S, T
S:=0; T:=V;
for all w in V do dist[w]:=infinity;
dist[u]:=0;

// add one vertex at a time to T
while T is non-empty do
**s:=vertex for which dist[s] represent the length
**   of the shortest path from u to s;
  add s to S; remove s from T;
  // update dist to account for enlarged set S
  for all t in out(s) do
    // try to shorten current path to t through s
    if (dist[s]+w[s,t]<dist[t]) then
      dist[t]:=dist[s]+w[s,t];
```

Ako vybrať vrchol s v kroku **?

Nech vrchol $s \in T$ má najmenšiu vzdialenosť $dist(s)$.
Potom $dist(s)$ je dĺžka globálne najkratšej cesty z u do s

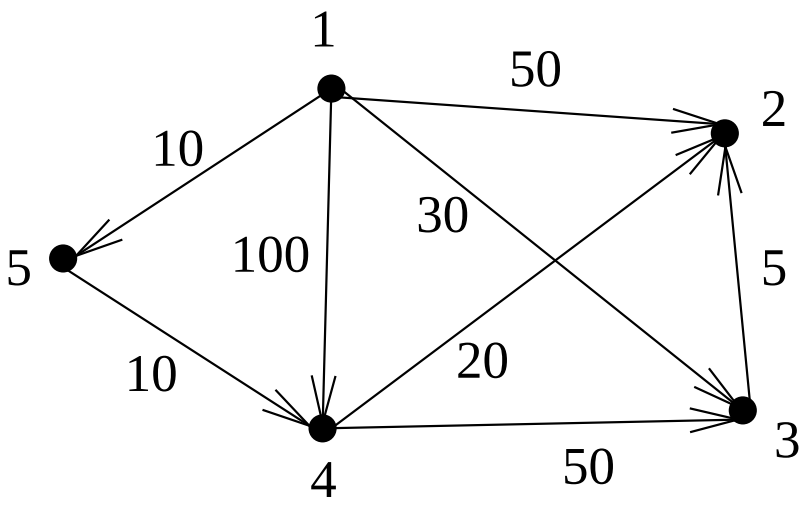
Predpokladajme, že z u do s existuje kratšia cesta.

Dijkstrov algoritmus

```
// initialize dist, S, T
S:=0; T:=V;
for all w in V do dist[w]:=infinity;
dist[u]:=0;

// add one vertex at a time to T
while T is non-empty do
  s:=vertex with the smallest dist[s];
  add s to S; remove s from T;
  // update dist to account for enlarged set S
  for all t in out(s) do
    // try to shorten current path to t through s
    if (dist[s]+w[s,t]<dist[t]) then
      dist[t]:=dist[s]+w[s,t];
```

Časová zložitosť: $\Theta(n^2)$ (pri triviálnej implementácii)



1	2	3	4	5
0	∞	∞	∞	∞

Dijkstrov algoritmus (s rekonštrukciou ciest)

```
// initialize dist, S, T
S:=0; T:=V;
for all w in V do
* dist[w]:=infinity; last[w]:=undefined;
dist[u]:=0;

// add one vertex at a time to T
while T is non-empty do
  s:=vertex with the smallest dist[s];
  add s to S; remove s from T;
  // update dist to account for enlarged set S
  for all t in out(s) do
    // try to shorten current path to t through s
    if (dist[s]+w[s,t]<dist[t]) then
*     dist[t]:=dist[s]+w[s,t]; last[t]:=s;
```

```
// path reconstruction from u to v
w:=v; create an empty path;
while last[w]<>undefined do
  add w to the beginning of the path;
  w:=last[w];
```

Ako je to s časovou zložitosťou?

$dist(s)$ nemusí byť pole; môže to byť iná dátová štruktúra:

- A: vytvor štruktúru s n prvkami
- B: zníž hodnotu konkrétneho prvku (použijeme m krát)
- C: vyber prvok s najnižšou hodnotou (použijeme n krát)

Celková časová zložitosť: $O(A + mB + nC)$

Implementácia poľom: $A = O(n)$, $B = O(1)$, $C = O(n)$
$\Rightarrow$ časová zložitosť $O(n^2)$

Implementácia haldou: $A = O(n)$, $B = O(\log n)$, $C = O(\log n)$
$\Rightarrow$ časová zložitosť $O(n + m \log n)$

Fibonacciho halda: $A = O(n)$, $B = O(1)$, $C = O(\log n)$
(využíva amortizovanú zložitosť)
$\Rightarrow$ časová zložitosť $O(m + n \log n)$

Zhrnutie

- Hľadanie najkratších ciest v ohodnotených grafoch
- Dijkstrov algoritmus: hľadá najkratšie cesty z jedného vrcholu do všetkých ostatných
- Vlastne sa jedná o zložitejší greedy algoritmus
- Časová zložitosť závisí od implementácie dátovej štruktúry
  triviálna $O(n^2)$ Fibonacciho halda $O(m + n \log n)$
- Nefunguje ak sú hrany záporné
- Na cvičeniach: Floyd-Warshallov algoritmus
  (príklad dynamického programovania)