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 $\Theta(n \log n)$
- 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:

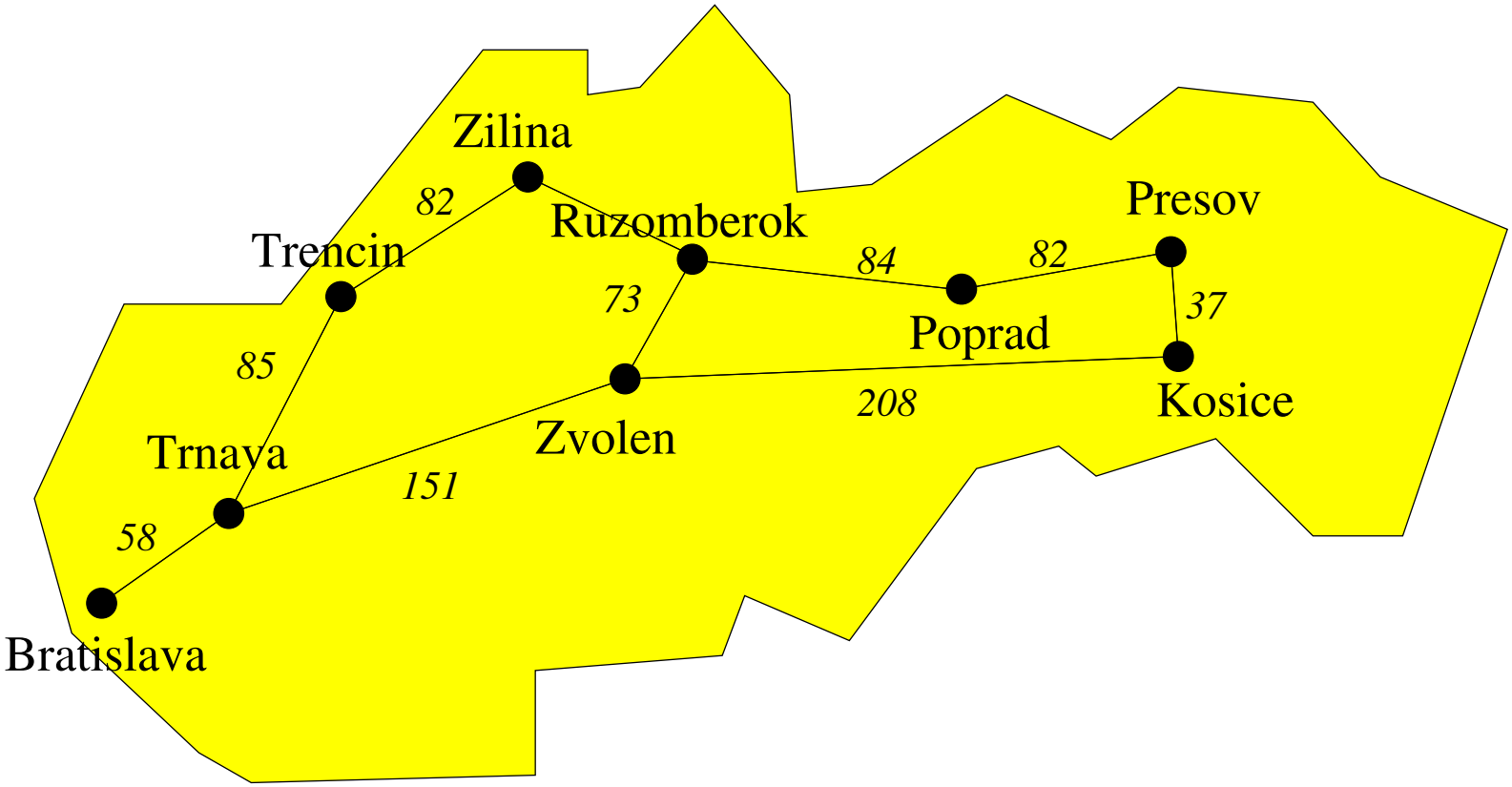
- Rozmieňanie peňazí (všeobecné) $\Theta(mS)$
- Celočíselný problém batohu $\Theta(nW)$
- Najdlhšia spoločná podpostupnosť $\Theta(mn)$
- Najkratšia triangulácia $\Theta(n^3)$

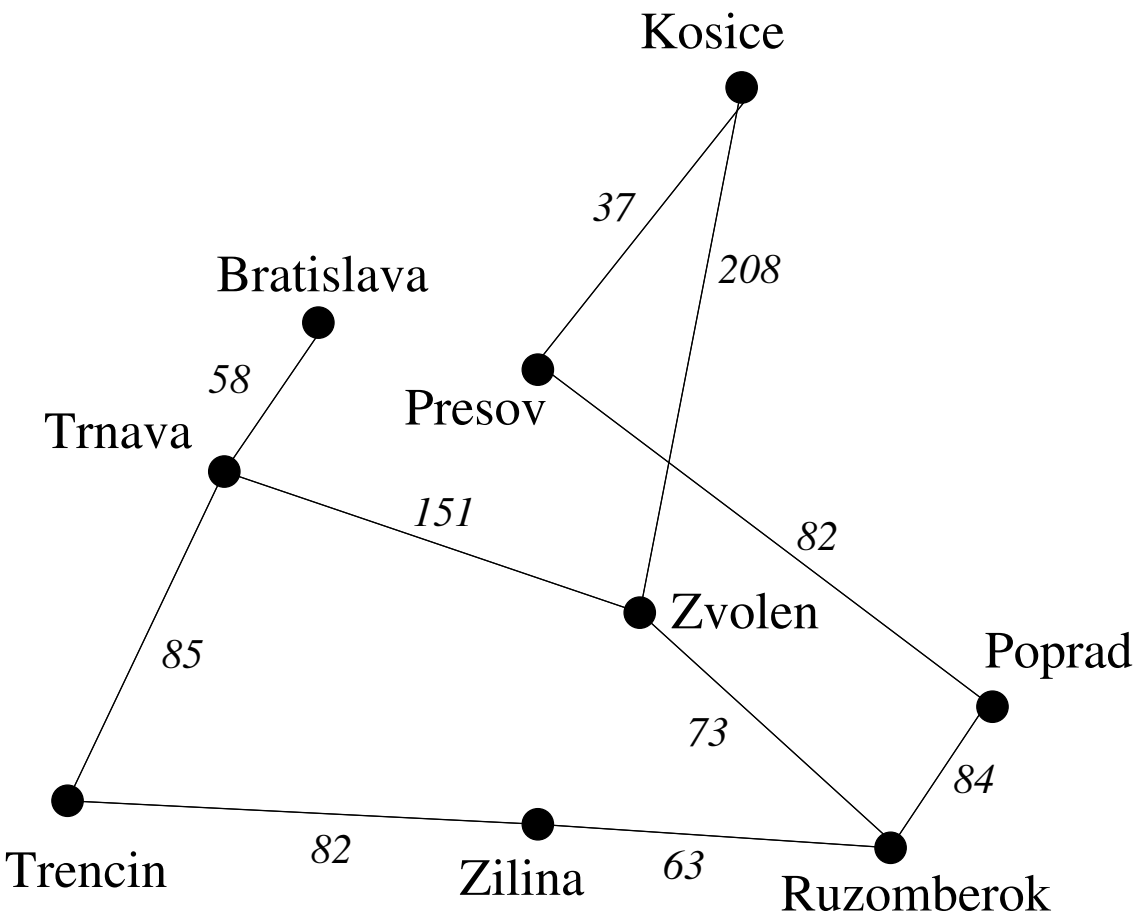
Opakovanie: Rozdeľuj a panuj

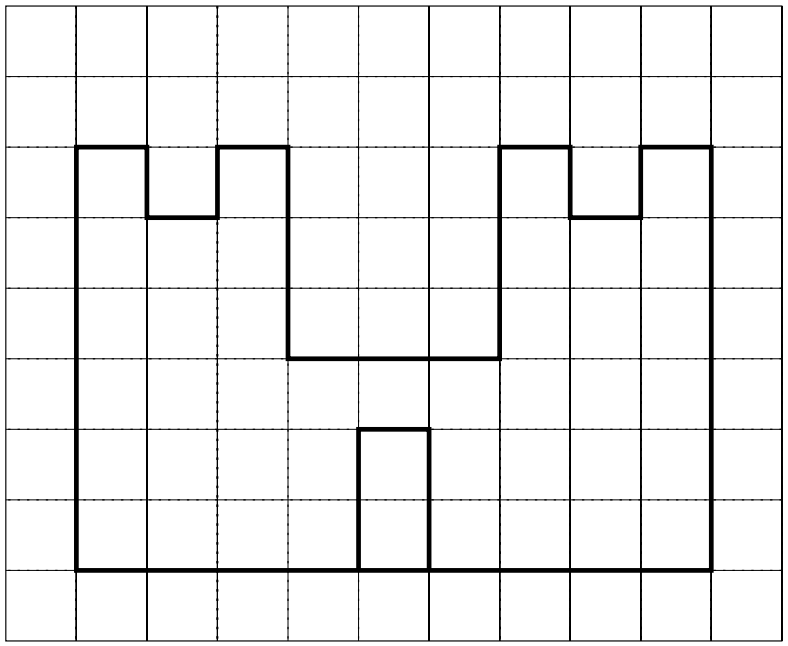
- rozdeľ problém na menšie podproblémy, vyrieš rekurzívne a skombinuj čiastkové riešenia
- niekedy ťažké na implementáciu, veľký overhead na rekurziu
- hlavný problém: analýza časovej zložitosti

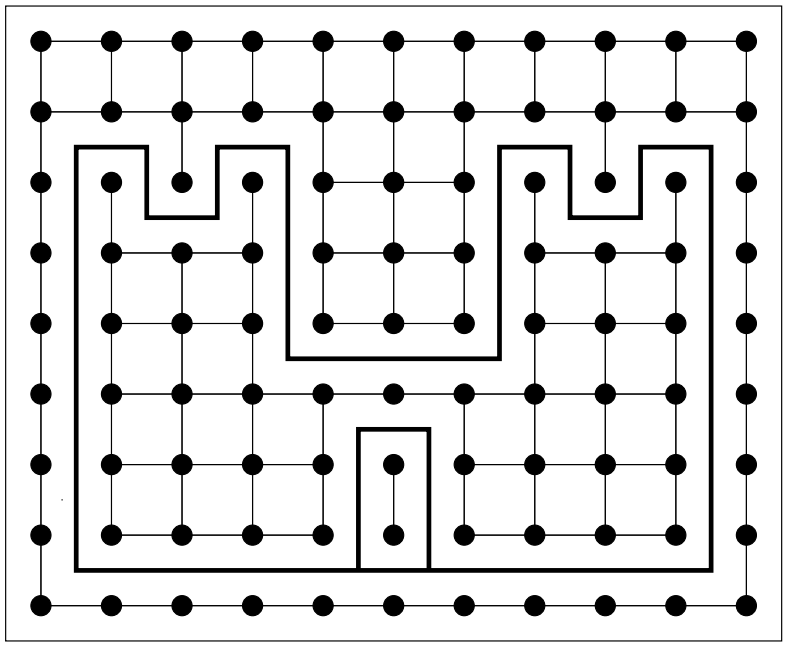
Príklady použitia:

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


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





Porovnanie reprezentácií grafov

	Matica susedností	Zoznamy susedov
$(u, v) \in E$?	$\Theta(1)$	$\Theta(\text{outdeg}(u))$
Hrany vychádzajúce z u	$\Theta(n)$	$\Theta(\text{outdeg}(u))$
Pamäť	$\Theta(n^2)$	$\Theta(m + n)$

```
function dfs-visit(v,cnum)
  // pre-condition: v is WHITE vertex
  // find all vertices that are reachable from v
  // by path going through white vertices only
  status[v]:=gray;
  num[v]:=cnum;
  for each w in out(v)
    if status[w]=white
       dfs-visit(w,cnum)
  status[v]:=black;

// --- main program ---
status of all vertices is white
cnum=0; // component number
for all vertices v in V
  if status[v]=white
    dfs-visit(v,cnum);
    cnum:=cnum+1;
```

```
function dfs-visit(v,cnum)
  status[v]:=gray;
* time:=time+1; d[u]:=time;
  num[v]:=cnum;
  for each w in out(v)
    if status[w]=white
*       edge (v,w) is a tree edge;
        dfs-visit(w,cnum)
  status[v]:=black;
* time:=time+1; f[u]:=time;
```

```
function bfs-visit(v,cnum)
  create empty queue Q;
  status[v]:=gray;
  dist[v]:=0;
  enqueue(Q,v);

  while Q is not empty
    u:=dequeue(Q);
    num[u]:=cnum;
    for each w in out(u)
      if status[w]=white then
        status[w]:=gray;
        dist[w]:=dist[u]+1;
        enqueue(Q,w);
    status[u]:=black;
```