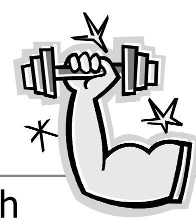
Algoritmické strategie I

I.Kolingerová

Obsah:

1. Brutální síla
2. Greedy (žravá, hltavá) strategie
3. Inkrementální techniky

1. Brutální síla



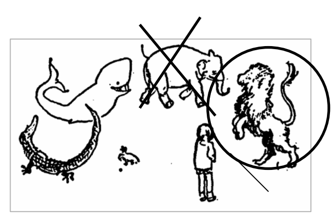
- Problém řešen prohledáním všech podmnožin, uspořádání nebo potencionálních řešení a výběrem nejlepšího
- Pro malé problémy postačující
- Často superpolynomiální složitost
- Obvykle snadná garance správnosti algoritmu

2

2. Greedy strategie

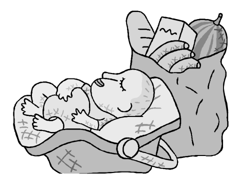
- Očekávané řešení: $(p_1,p_2,\ldots,p_n)$
- Začít s ø
- Zafixovat vždy 1 položku řešení (např. p_2), už zůstane neměnná, pak další položku (např. p_1) atd., až vše pevně nastaveno
- Pořadí podle slibnosti položek (od slibnějších k méně)
- "greedy" – nikdy se nevrací, nikdy nemění dříve udělané rozhodnutí

- Př.: Vybrat nejatraktivnější zvířata do ZOO



- Některé položky se vzájemně vylučují - ale nedělá se žádná predikce, žádný návrat

- Realizace jedné cesty od kořene k uzlu v rozhodovacím stromě
- Obvykle velmi účinné algoritmy
- Greedy algoritmy – snadný vývoj i implementace
- Obvykle neumí najít globální extrém, ale poskytne dobrou heuristiku

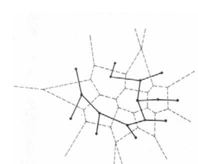


Př.1: Greedy algoritmus pro minimální kostru (minimum spanning tree - MST)

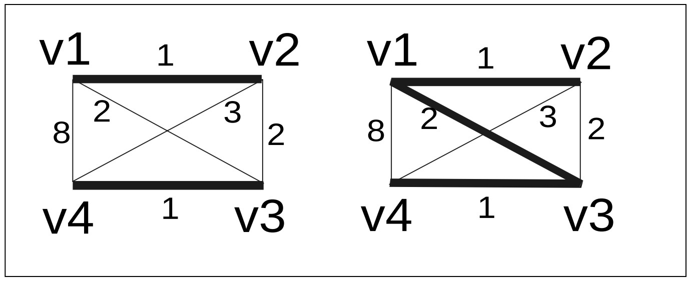
- Input: Ohodnocený spojitý graf
  G=(V,E,c),c:E->N

účinný alg.

- Output: (V,E´), kde E' - min. kostra
- Step 1: Seřaď hrany podle jejich ohodnocení:
  $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$
- Step 2: Set E´:=$\{e_1,e_2\}$,i:=3;
- Step 3: **while** |E'|<|V|-1 **do**
- **begin**
- Přidej e_i do E' pokud
  (V,E'u$\{e_i\}$) neobsahuje smyčku;
- i:=i+1
- **end**



greedy strategie pro MST



- Jaká je složitost nejhoršího případu vzhledem k počtu hran |E| ?
- Jak implementovat krok 3 v O(1) nebo O(log *n*) na jednu hranu?

7

Př.2: Greedy heuristika pro minimálního cestujícího (travelling salesperson - TSP)

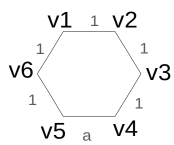
greedy strategie pro TSP

velmi slabé

- Input: Ohodnocený úplný graf
  G=(V,E,c),c:E->N, |V|=*n, n* z N^+
- Output: (V,E´), kde E' - min. cesta
- Step 1: Seřaď hrany podle jejich ohodnocení:
  $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$, $m = \binom{n}{2}$
- Step 2: Set E´:={e_1,e_2},i:=3;
- Step 3: **while** |E'|<*n* **do begin**
- Přidej e_i do E' pokud
  (V,E'u{e_i}) neobsahuje vrchol st. > 2
  ani smyčku délky kratší než n;
- i:=i+1
- **end**

8

- Nalezené řešení může být jakkoliv horší než optimum (žádná zaručená mez kvality)

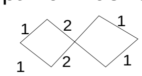


ostatní hrany váhu 2

a >> 1

Greedy-TSP vezme
napřed hrany délky 1
=> délka a+5

Optimum: délka 8



3. Inkrementální strategie

a) Inkrementální vkládání
b) Inkrementální výběr (konstrukce)
c) Jiné inkrementální strategie -
    např. inkrementální změna

- Obvykle nevede k optim. algoritmům
- Obvykle implementačně jednoduché, zejména vkládání
- Dovoluje různé modifikace

a) Inkrementální vkládání

- Budujeme řešení (strukturu) o *n* položkách - napřed řešení (struktury) o *n*-1 položkách, pak nezbytné změny a vložení *n*-té položky
- V paměti někdy dílčí řešení, někdy jen současný stav
- Zvlášť užitečné pro geometrické algoritmy
- On-line metoda - vstup nemusí být celý dostupný na začátku - užitečné pro reálná data, ale část práce se dělá zbytečně

11

Př.1: Insertion_sort

inkrement. vkládání pro řazení

- Input: pole neseřazených hodnot A[1..n-1]
- Output: seřazené pole A
- Step 1: A[0] := -∞ // "zarážka (sentinel)"
- Step 2: **for** i := 1 **to** *n*-1 **do**
- **begin**
- j:=i;
  while (A[j]<A[j-1]) **do begin**
- Swap (A[j,]A[j-1]); j:=j-1 **end;**
- **end**
- $O(n^2)$ v nejhor. př., pokud data skoro seřazena, podstatně lepší

12

Př.:

n=5 $-\infty, 1, 5, 2, 4$

i	j				
1	1	$1 < -\infty$	F		
2	2	$5 < 1$	F		
3	3	$2 < 5$	T	Swap(5,2)	$-\infty, 1, 2, 5, 4$
3	2	$2 < 1$	F		
4	4	$4 < 5$	T	Swap(5,4)	$-\infty, 1, 2, 4, 5$
4	3	$5 < 2$	F		

Př.3: Obarvení vrcholů

inkrement. vkládání pro obarvení

- Zadání: Obarvěte vrcholy grafu minimálním počtem barev tak, aby žádná hrana nespojovala vrcholy stejné barvy
- Aplikace: shlukování, plánování, optimalizace překladačů (užití koneč. počtu registrů - proměnné překrývající se v čase nemohou být ve stejném registru)
- Počet potřebných barev grafu - chromatické číslo, jeho určení je NP-úplný problém, pro planár.graf nejvýše 4
- => pro přesné určení pro malé grafy backtracking - pro malé náhodné grafy funguje překvapivě dobře

inkrement. vkládání pro obarvení

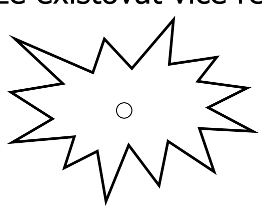
- Heuristika: obarvovat vrcholy sekvenčně podle omezení daných už obarvenými
- Pořadí vrcholů: vkládat vrcholy v pořadí podle jejich stupně (sestupně) <= vyšší stupeň- více omezení
- Vylepšení: výměna barev - zrušíme v obarvení všechny vrcholy kromě červených a modrých, pokud se rozpadne na 2 a více komponent, lze přebarvit červenou na modrou (nebo naopak ;-) a pokračovat v barvení => růst kvality řešení za cenu růstu doby výpočtu a složitosti implementace

15

inkrement. vkládání pro hvězdu

Př.4: Hvězdicová polygonizace

- Zadání: sestrojte hvězdicovou polygonizaci množiny bodů S (může existovat více řešení)



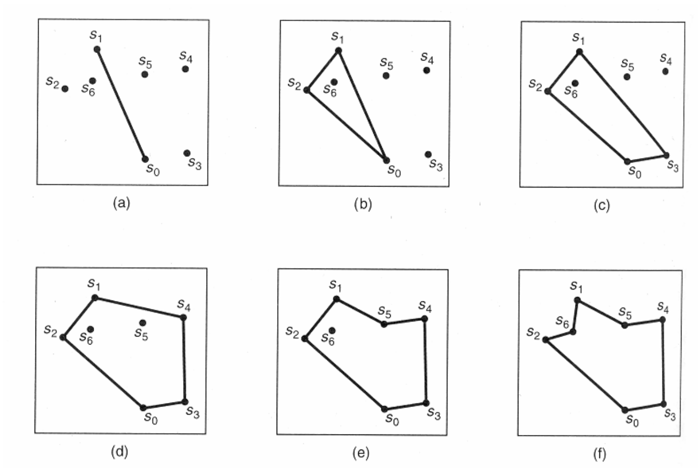
Algoritmus:

- Iterativní konstrukce polygonu P z S, na poč.P=(s_0)
- V každé iteraci i od 1 do *n*-1 je do P na správné místo podle úhlu zařazen další bod s_i

16

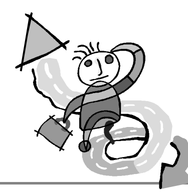
- Průchod po směru hodin. ručiček od počát. vrcholu s_0 až do vrcholu, kt. se stane následníkem s_i; s_i je pak vloženo před tento vrchol (s_0-zarážka)
- Start z jiného vrcholu může dát jiné řešení

O(n²)



17

b) Inkrementální konstrukce



- Budujeme řešení (strukturu) inkrementálně, v každém okamžiku "kousek"
- Vstup zpracováván v pořadí, jaké algoritmus chce, nikoliv v pořadí, jak vstup. data přicházejí - off-line - vstup musí být celý k dispozici
- Vhodné pořadí vstup. dat někdy určeno předem (např. presort), někdy až v průběhu výpočtu
- Časté pro geometrické problémy

18

Př.1: Selection_sort

inkrementální konstrukce

- Algoritmus: opakovaně vybírá nejmenší položku z množiny, až je množina prázdná (v poli: vyměnit min s 1. prvkem atd.)
- $O(n^2)$, ale jen n výměn - insertion_sort kolem $n^2/2$ polovýměn (posunů) v nejhor. případě

- Úkol k procvičení: Udělejte alg. pro hvězdic. polygonizaci inkrementální konstrukcí

19

c) Jiné inkrementální strategie

- Znám už jen jednu ☺ - inkrementální změna
- Př.: Generování permutací - generujte všechny/náhodnou/další permutaci délky n
- Permutace v poli, inkrement. změna = jeden swap v poli
- Inkrem. změna generuje další permutaci z předchozí - extrémně rychlé - až O(1) v průměru pro 1 permutaci - nezávisí na n !
- Velmi "tricky" - viz cvičení

20