

Zábavné algoritmy

I.Kolingerová

THINGS
AREN'T
ALWAYS
#000000
AND
#FFFFFF

Obsah:

1. Pessimální algoritmy a složitostní analýza
2. Bizarní řídicí algoritmy
3. Další užitečné techniky
4. Rekreační informatika



Pess

Sim

1. ~~Optimal Algorithms and Complexity Analysis~~

- *Účel:* Procvičení algoritmického myšlení obrácením obvyklého přístupu naruby
- **Problém 1:** hledáme X v poli n celých čísel A_1, A_2, \dots, A_n co nejpomaleji; sekvenční hledání nestačí - $O(1)$ v nejlepším případě, chceme lepší, tj. horší řešení
- Triviální řešení: přidat na začátek prázdné smyčky – příliš okaté
- Lepší: A seřadit vzestupně, nasadit tzv. zdráhavé prohledávání
- *Def.:* Zdráhavý algoritmus je algoritmus, který maří čas způsobem dostatečně vynalézavým pro oklamání naivního pozorovatele

[BS84]

Procedure *reluctant_search* (X, i, j : integer): integer

// vrátí index k , pokud $A_k = X$, nebo -1, pokud takové k neex.

if $i > j$ **then** return -1 **endif**

if $i = j$ **then if** $X = A_i$ **then** return i **else** return -1 **endif**

$m \leftarrow \text{trunc}(i+j)/2$

if $X \leq A_m$ **then**

$k \leftarrow \text{reluctant_search}(X, m+1, j)$

if $k = -1$ **then** return *reluctant_search* (X, i, m) **else** return k **endif**

else

$k \leftarrow \text{reluctant_search}(X, i, m)$

if $k = -1$ **then** return *reluctant_search* ($X, m+1, j$) **else** return k **endif**

endif

endproc

Problém 2: Hledání cesty v příjemném bludišti

- Dáno bludiště v podobě grafu G s n uzly a vstupem v uzlu u , chceme najít výstup v uzlu v , ale chceme si to v bludišti užít (spěchej pomalu)
- Sloppiest-path algoritmus [Hom]: navštívit skoro všechny uzly grafu bludiště – metoda převráceného gradientu/nejmenšího spádu – tzv. Penelopina strategie – užívá **for** smyčku, jejíž krok v každé iteraci osciluje mezi kladným a záporným krokem
- Tento alg. bohužel dnes obecně znám jako tzv. backtracking

Problém 3: očíslování všech uzlů ve spojitém grafu

- Klasicky se řeší algoritmem DFS [Ve15] nebo BFS [Ve25] v $\Omega(n)$
- Odborná komunita zdrženlivých algoritmů vyvinula tzv. backwards-first search

Procedure *bwfs* (*v* : vertex, *i*: integer)

// předpokl. počáteční ohodnocení uzlů 0, ohodnotí 1 až n

$\Omega(n^2)$,
best case $O(n^{3/2})$

l(v) <- i // do l(.) se uloží ohodnocení vrcholů

for each neighbor *u* of *v* **do**

if $0 < l(u) < i$ **then** *bwfs* (*u*, *i*) **endif**

endfor

for each neighbor *u* of *v* **do**

if $l(u) = 0$ **then** *bwfs* (*u*, *i+1*) **endif**

endfor

endproc

Problém 4: Zdráhavé řazení

- Seřadíte prvky v poli nebo části pole $A[l\dots r]$
- Strategie multiply and surrender:
 - nahradí problém dvěma nebo více jednoduššími podproblémy
 - vytváří násobné množství podproblémů, jak dlouho to jen jde
 - když už řešení podproblému pro svoji jednoduchost nejde odkládat, vzdá se a problém vyřeší
- Na tom založen Slowsort (podrobně později):
 - Nalezení maxima
 - Rekurzivní řazení dvou částí pole
 - Rekurzivní řazení celého pole kromě maxima
- Využití: šéf nás pošle něco seřadit v Paříži a platí pobytové výdaje

2. Bizarní řadící algoritmy

- Účel: Zábava a lepší pochopení algoritmizace, nikoliv praktické využití
- Vstup: pole prvků

a) *Bogosort*

- Pokud nejsou prvky seřazené, opakovat: seřadit prvky do náhodného pořadí
- Nejhorší případ: $O(\infty)$, nejlepší $O(n)$, průměr $O((n+1)!)$

b) *Bogobogosort*

- Zkontrolovat seřazení prvních dvou prvků a bogosortovat je
- Dtto první 3 prvky atd.
- Pokud prvky stále nejsou seřazené, opakovat s 2 prvky atd.
- $O(n! * 1! * 2! * n!)$

c) *Sleepsort*

- Pro každý prvek v poli vygenerovat vlákno, které čeká po dobu odpovídající jeho hodnotě, potom pošle dané číslo na výstup
- Jen pro kladná čísla

d) *Kvantový bogosort*

- Předpoklad: existuje nekonečné množství paralelních vesmírů, v jednom z nich jsou prvky seřazené
- Postup:
 - Zamíchat prvky do náhodného pořadí
 - Pokud nejsou seřazené, zničit současný vesmír a vše opakovat v jiném

e) *Slowsort*

- Parodie na D&C algoritmy, nešikovné využití rekurze bez snahy o její efektivitu
- Postup:
 - Rekurzivně dělit datovou strukturu s n prvky na 2 menší části (s užitím „prostředního“ indexu), konec, když levý index \geq pravý (jako půlení intervalu)
 - Když jsou prvky už jen 2, dát větší prvek jako druhý
 - Potom rekurze na $n-1$ prvcích (bez maxima)
- Složitost: $O(n^{O(\log n)})$

Složitost: $O(n^{O(\log n)})$

Procedure *slowsort* ($A, l, r : \text{integer}$)

// Seřadí prvky A[l...r] vzestupně

if $l \geq r$ **then** return

else

$m \leftarrow \text{trunc}(l+r)/2$

slowsort (A, l, m)

slowsort ($A, m+1, r$)

if $A_m > A_r$ **then** $A_m \leftrightarrow A_r$ **endif**

slowsort ($A, l, r-1$)

endif

endproc

f) *Purgesort (též Stalinův sort)*

- Procházet prvky zleva doprava
- Zachovat jen prvky, které jsou ve správném pořadí
- Výstup – seřazený za cenu ztráty dat
- $O(n)$

g) *Komunistický sort*

- Nahradit všechny prvky jejich průměrem
- $O(n)$

h) *Miraclesort*

- Předpokládáme, že prvky jsou seřazené
- Pokud někdo nesouhlasí, trváme na to, že by nám měl více věřit

3. Další užitečné techniky

a) *Nejkratší důkaz libovolné hypotézy* [Fer]

Př.: důkaz existence lochnesské příšery

Výrok: Existuje lochnesská příšera, nechť P je pravděp. tohoto výroku,
nechť p je pravděp. negace tohoto výroku

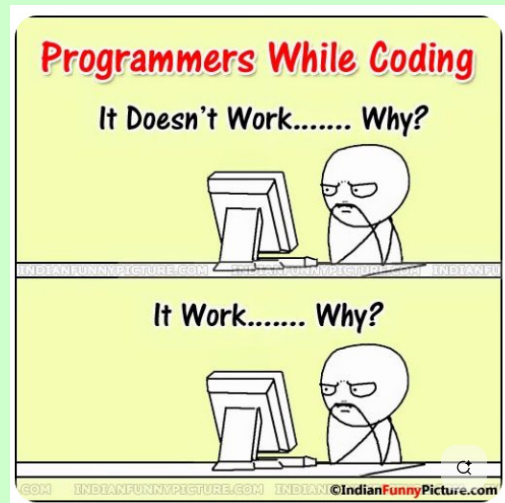
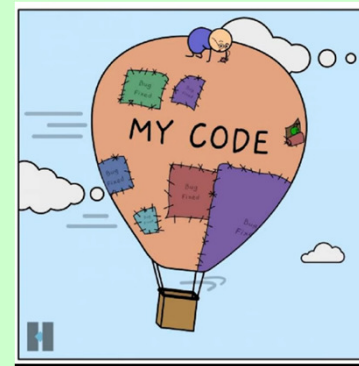
Důkaz: každý ví, že $P > p$, čímž je důkaz proveden

Pozn.: pozor na záměnu značení 😊

b) *Proces vývoje software*

1. Objednat trička pro vývojový tým.
2. Ohlásit dostupnost daného sw.
3. Napsat kód.
4. Napsat manuál.
5. Najmout PM (produktového managera).
6. Napsat specifikaci sw (psaní podle hotového sw => splnění požadavků na sw zajištěno).
7. Prodávat sw.
8. Otestovat sw (s využitím uživatelů).
9. Identifikovat chyby jako potenciální rozšíření sw.
10. Ohlásit upgrade.

c) Mozaika ze života programátora



4. Rekreační informatika

a) Přesouvání mincí

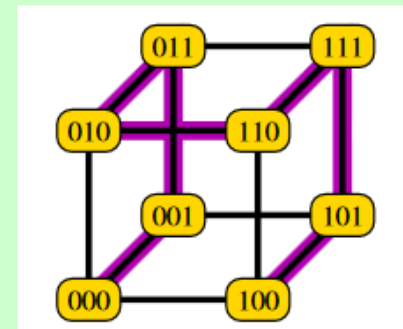
[BDD08]

Divák uspořádá 3 mince do řady, libovolnou stranou vzhůru, ale ne všechny stejně. Kouzelník konfiguraci nevidí, přesto je max. 3 tahy převrátí všechny stejnou stranou vzhůru.

Řešení: obrátit levou, if not OK, prostřední, if not OK, znovu levou

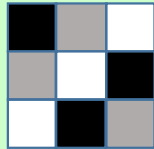
0 0 1 → 1 0 1 → 1 1 1 OK
0 1 0 → 1 1 0 → 1 0 0 → 0 0 0 OK
0 1 1 → 1 1 1 OK
1 0 0 → 0 0 0 OK
1 0 1 → 0 0 1 → 0 1 1 → 1 1 1 OK
1 1 0 → 0 1 0 → 0 0 0 OK

- Proč to funguje?
- Souvislost: Grayovy kódy ukazují cestu
- Konfigurace mincí – vrchol 3D krychle
- 2^3 konfigurací, ale stačí 1/2 (2 cílové stavy) – vlastně jen čtverec
- Max. 3 změny mezi 4 konfiguracemi
- Pro 2^n konfigurací je 2^{n-1} dvojitych konfigurací, $2^{n-1} - 1$ přesunů v nejhorším případě



b) Hra Tripas – výzva pro vás

[FGD04]



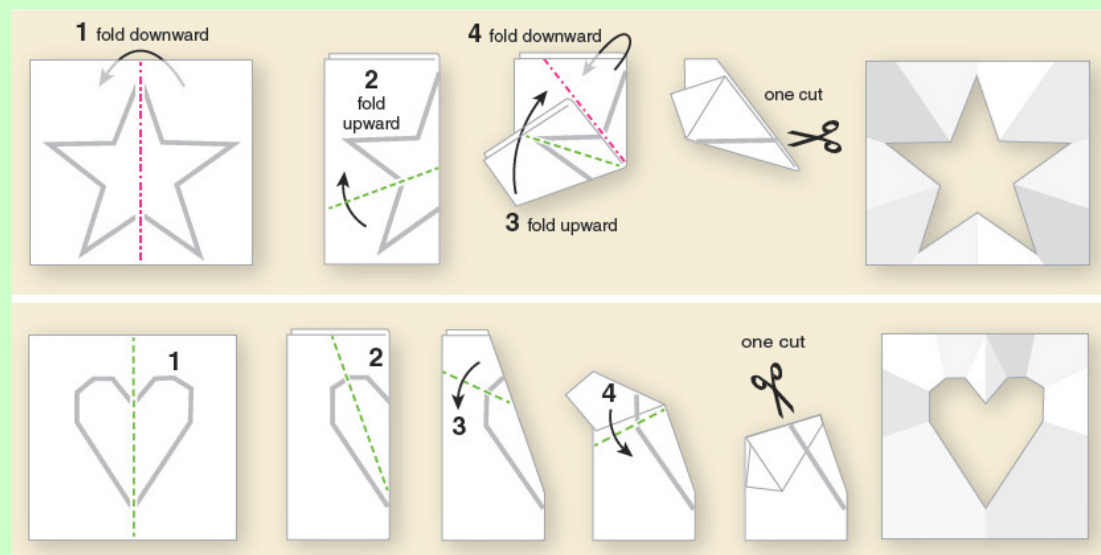
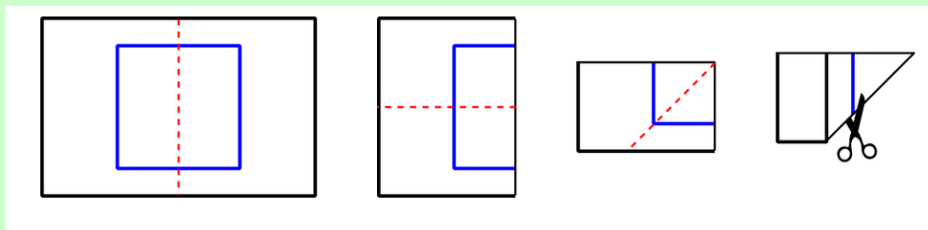
- Hra pro 2 osoby, každý má 4 kameny, jeden bílé, jeden černé
- Střídavě pokládají kameny na prázdná pole
- Vítězí hráč, který umístí 3 symboly v řádku, sloupci nebo na polích stejné barvy
- Pokud neuspěje první hráč ve svých 4 tazích, vítězí druhý
- Existuje vítězná strategie pro prvního hráče, dokážete ji najít?

c) *Jedním řezem – ještě větší výzva pro vás*

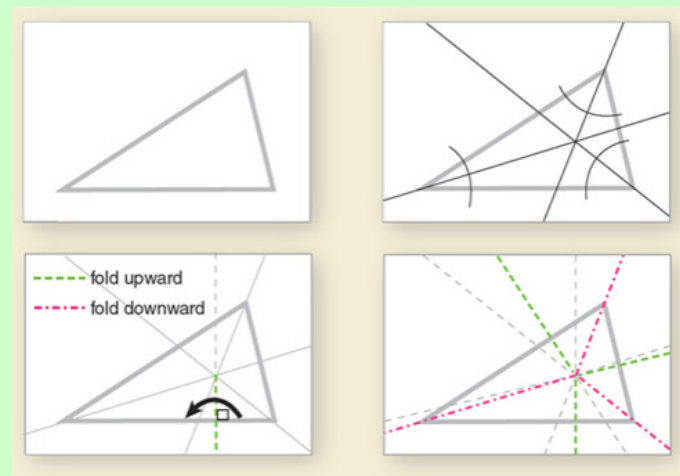
[Dem12, Dem25]

Je možné získat i složité tvary skládáním papíru a jedním řezem

Inspirační příklady:



- Strany tvaru, který chceme vyříznout, by se měly dostat do jedné přímky (v jednodušším případě je to osa zrcadlové symetrie daného tvaru)
- Polygon jde vyřešit vždy, více polygonů, popř. obecný graf mohou přinést zapeklité situace
- Sklady mohou být složité, viz např. trojúhelník – využity osy úhlů a střed kružnice vepsané, skládá se nahoru i dolů



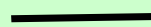
- Existují dvě řešení :

- 1. Využití kostry a kolmic stran – praktičtější, funguje „skoro vždy“

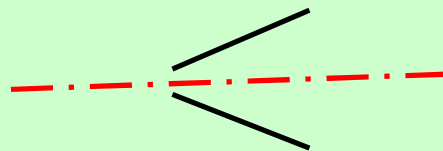
- 2. Využití balení disků – teoretičtější, složitější, funguje vždy

- Kostra:

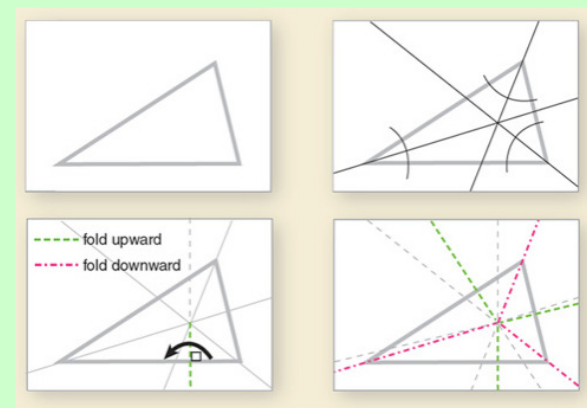
- 1 úsečka – neděláme nic



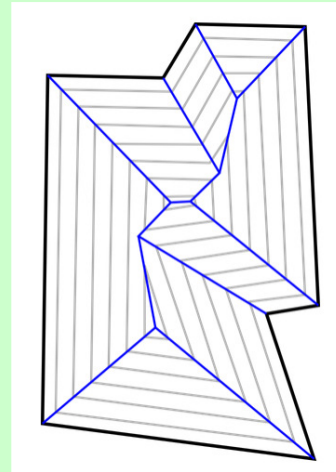
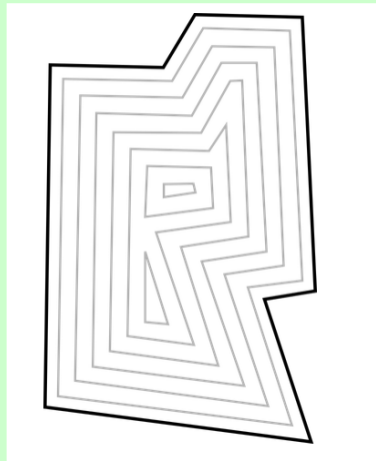
- 2 úsečky – potřebují srovnat do přímky – osa, v ní sklad



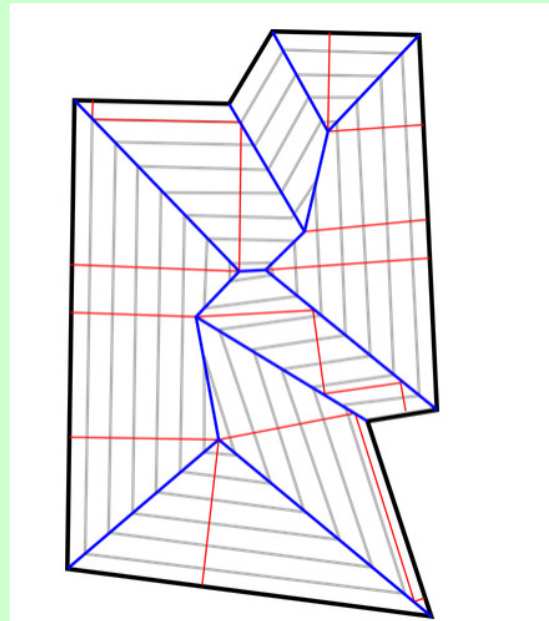
- Trojúhelník – osy úhlů a kolmice stran



- Obecnější tvary: hrany kostry jsou vlastně trajektorie vrcholů daného tvaru při jeho zmenšování

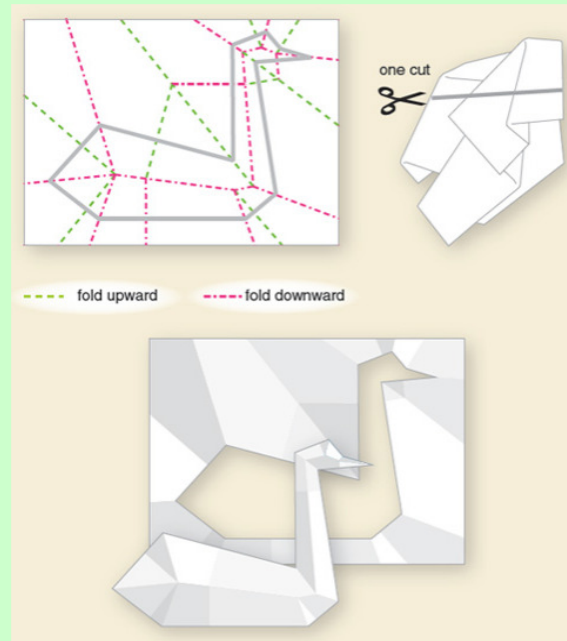


- Z vrcholů kostry se spustí kolmice k hranám původního tvaru, pokud tyto spojnice někde protnou kostru, „odrazí se“ (ve stejném úhlu jako dopadly), pokračuje se, až protnou hranu původního tvaru v pravém úhlu



- Pro skládání se využije kostra a některé z kolmic

- Složitější ukázka:



- Překvapující aplikace pro skládání 3D ploch, např. simulace rozvinutí airbagu kompaktně složeného do malého prostoru
- Výzva pro vás: výběr skladů a jejich pořadí – aspoň poloautomatické řešení

Závěr

- Do informatiky patří i specifický programátorský humor, není zdravé, když člověk sám sebe bere smrtelně vážně
- I rekreační a bizarní témata mohou poskytnout vhodná témata pro zábavný trénink algoritmického myšlení

Literatura (1)

- [BDD08] Benbernou N., Demaine E. D., Demaine M. L., and Rossman B.: Coin-Flipping Magic, in Exchange Book of the 8th Gathering for Gardner (G4G8), Atlanta, Georgia, March 2008.
- [BS84] Broder A., Stolfi J.: Pessimal Algorithms and Simplicity Analysis, SIGACT News 16, 3 (Fall 1984), 49–53, <https://doi.org/10.1145/990534.990536>
- [Dem12] Demaine E.: Class and Lecture Videos, Lecture 8, Fold & One Cut, 2012, <https://ocw.mit.edu/courses/6-849-geometric-folding-algorithms-linkages-origami-polyhedra-fall-2012/resources/lecture-8-fold-one-cut/>
- [Dem25] Demaine E.: Erik Demaine's [Folding and Unfolding](https://erikdemaine.org/foldcut/): The Fold-and-Cut Problem, 2025
- [FGD04] Farrell J., Gardner M., Rodgers T.: Configuration Games, in: Cipra B., Demaine E.D., Demaine M.L., Rodgers T.: Tribute to a Mathemagician, CRC Press, 2004
- [Fer] Ferko, A.: Najkratší dôkaz hocičoho, https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.youtube.com%2Fwatch%3Fv%3Df0_YbMk9-eY&psig=AOvVaw0V99tQl-McOY4CR19IacQ5&ust=1756461155006000&source=images&cd=vfe&opi=89978449&ved=0CBUQjhxqFwoTCPDA9OyerY8DFQAAAAAdAAAAABAL
- [Gud24] Gudzik I.: The Most Bizzare Sorting Algorithms, 2024, <https://codefinity.com/blog/The-Most-Bizarre-Sorting-Algorithms>
- [Hom] Homér: Odyssea

Literatura (2)

- [Sort1] <https://www.youtube.com/watch?v=ktgxMtWMfIU>
- [Sort2] <https://www.youtube.com/watch?v=Cp9mdJmVtvo>
- [Ve15] Verne J.: Cesta do středu Země, např. nakladatelství Josef Vybíral, Žalkovice 2015, přeložil J. Wagner, znovu 2023.
- [Ve25] Verne J.: Cesta kolem světa za osmdesát dní, např. nakladatelství Dobrovský, Praha 2025, přeložila Lucie Secká