Informatika pre biológov

Broňa Brejová

27.9.2018

Formulácia problému, algoritmus

- **Formulácia problému:** jasne zadefinujeme vstupné a výstupné dáta a aký výstup očakávame pre každý vstup.
- Vo formulácii nehovoríme **akým spôsobom** vypočítame výstupy zo vstupov.
- **Správny algoritmus:** Postup, ktorý určuje spôsob, akým pre každý vstup vypočítame príslušný výstup.

Biologický problém: Pomocou hmotnostného spektrometra (mass spectrometer) sme odmerali vo vzorke peptid s hmotnosťou K. Máme databázu proteínov a chceme zistiť, ktorý z proteínov obsahuje peptid s touto hmotnosťou.

Informatický problém: Vstup je postupnosť n kladných čísel $a[1], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Príklad:

```
K=19
3 4 6 3 6 4 9 2 8
    ^^^^^^^
```

Informatický problém: Vstup je postupnosť n kladných čísel $a[1], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Triviálne riešenie: skúšame všetky možnosti

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |  suma := suma + a[u]
|   |   ak suma = K, vypíš i,j
```

```
K=19
3 4 6 3 6 4 9 2 8
    i     j
```

Ako dlho takýto program pobeží?

- Naimplementovať do počítača a odmerať
- Na akom počítači? Na akých vstupoch?
- **Časová zložitosť:** (označujeme $O(f(n))$)
  - Zvolíme si parameter charakterizujúci množstvo dát
    napr. počet prvkov vstupnej postupnosti n
  - Pre každú veľkosť vstupu **odhadneme najhorší prípad**
  - Zanedbáme konštanty

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   suma := 0;
|   |   pre každé u od i po j
|   |   |  suma := suma + a[u]
|   |   ak suma = K, vypíš i,j
```

Časová zložitosť: kubická, alebo $O(n^3)$

Prečo je časová zložitosť dôležitá a konštanty nie?

		$O(n)$	$O(n \log n)$	$O(n^2)$	$O(n^3)$	$O(2^n)$
Čas na	10	ε	ε	ε	ε	ε
vyriešenie	50	ε	ε	ε	ε	2 weeks
problému	100	ε	ε	ε	ε	2800 univ.
veľkosti	1000	ε	ε	0.02s	4.5s	—
...	10000	ε	0.01s	2.1s	75m	—
	100000	0.04s	0.12s	3.5m	52d	—
	1 mil.	0.42s	1.4s	5.8h	142yr	—
	10 mil.	4.2s	16.1s	24.3d	140000yr	—
Max veľkosť	1s	2.3 mil.	740000	6900	610	33
problému	1m	140 mil.	34 mil.	53000	2400	39
vyriešená za	1d	200 bil.	35 bil.	2 mil.	26000	49
Zvýšenie	$+1$	—	—	—	—	$\times 2$
času so	$\times 2$	$\times 2$	$\times 2+$	$\times 4$	$\times 8$	—
zvýšeným n						

Efektívnejší algorimus

- Najprv si predpočítame pre každý začiatok postupnosti jej súčet $S[i] = a[1] + a[2] + \cdots + a[i]$

```
S[0] := 0
pre každé i od 1 po n
|   S[i] = S[i-1] + a[i]
```

```
a: 3  4  6  3  6  4  9  2  8
S: 3  7 13 16 22 26 35 ...
```

- Potom súčet podpostupnosti od i po j vieme spočítať jednoducho ako $S[j] - S[i-1]$

```
pre každé i od 1 po n
|   pre každé j od i po n
|   |   ak S[j]-S[i-1] = K, vypíš i,j
```

- **Časová zložitosť:** kvadratická, alebo $O(n^2)$
- Ak sú všetky čísla kladné, dá sa aj v lineárnom čase $O(n)$

Ďalší príklad informatického problému z prednášky

Najkratšie spoločné nadslovo

- Vstup: niekoľko reťazcov
- Výstup: najkratší reťazec, ktorý obsahuje všetky vstupné reťazce ako súvislé podreťazce

Príklad:

Vstup: GCCAAC, CCTGCC, ACCTTC

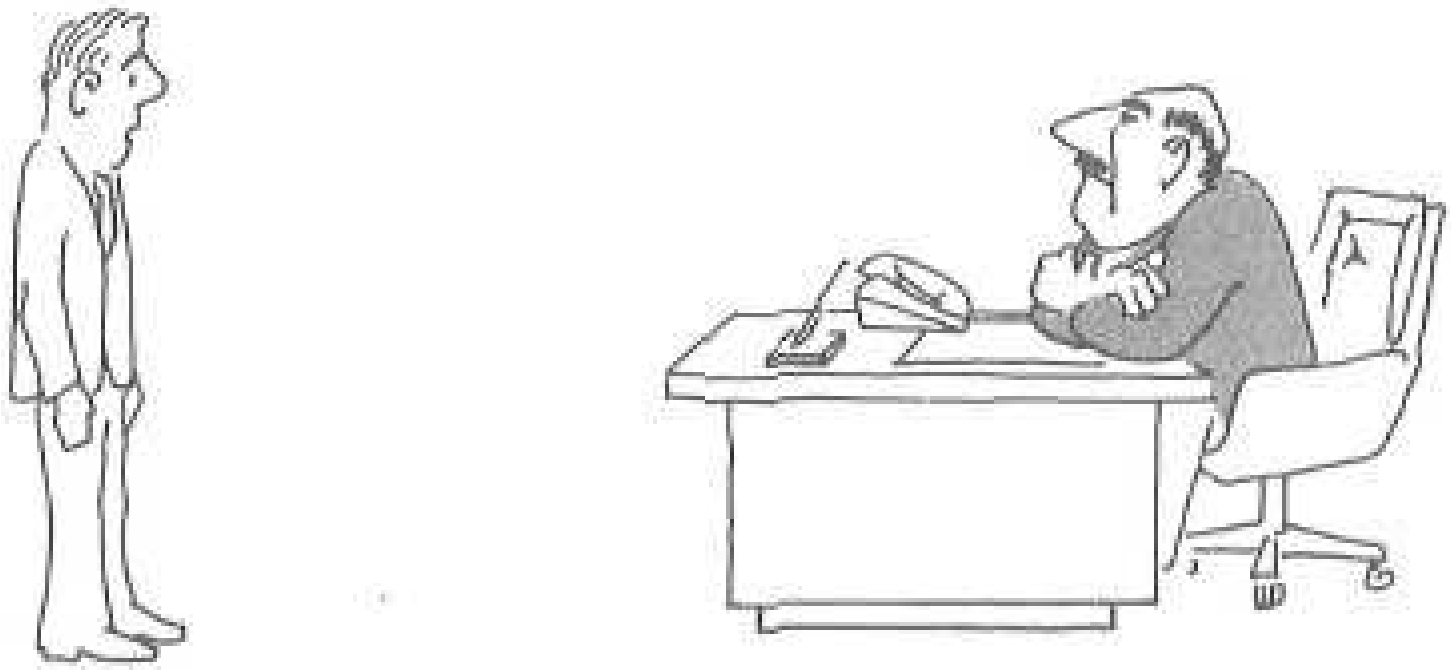
Výstup: CCTGCCAACCTTC (najkratšie možné)

Ako rýchly algoritmus poznáme pre tento problém?

Najkratšie spoločné nadslovo

Nepoznáme algoritmus, ktorý by bežal v polynomiálnom čase t.j. $O(n^k)$ pre nejakú konštantu k

Tento problém je **NP-ťažký**.



"I can't find an efficient algorithm, I guess I'm just too dumb."

"I can't find an efficient algorithm, because no such algorithm is possible!"

"I can't find an efficient algorithm, but neither can all these famous people."

Ako sa vysporiadať s NP-ťažkými problémami?

Heuristické algoritmy

- Nájde **aspoň nejaké riešenie**, aj keď nie nutne optimálne
- Nejde teda o správny algoritmus riešiaci náš problém, lebo pre niektoré vstupy dáva zlú odpoveď
- Radšej ale horšia odpoveď rýchlo, ako perfektná o milión rokov

Príklad: Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Príklad: CATATAT, TATATA, ATATATC
Optimum: CATATATATC, dĺžka 10
Heuristika: CATATATCTATATA, dĺžka 14

Ako so vysporiadať s NP-ťažkými problémami?

Aproximačný algoritmus

- Často vieme dokázať, že nejaká heuristika sa vždy priblíži k optimálnemu riešeniu aspoň po určitú hranicu

Príklad: Heuristika pre najkratší spoločný nadreťazec: v každom kroku zlepíme dva reťazce s najväčším prekryvom

Je dokázané, že vždy nájde najviac 3,5-krát dlhší reťazec ako najlepšie riešenie.

Informatici predpokladajú, že v skutočnosti najviac 2-krát dlhší, ale nevieme to dokázať.

Ako so vysporiadať s NP-ťažkými problémami?

Exaktný výpočet pomocou iného problému

- Preformulovať do podoby jedného z dobre známych NP-ťažkých problémov (napr. celočíselné lineárne programovanie, a pod.)
- Múdri ľudia napísali programy, ktoré vedia riešiť tieto známe problémy **aspoň v niektorých prípadoch** (CONCORD, CPLEX, a pod.)

Preformulovať problém

- Je toto skutočne jediná rozumná formulácia biologického problému ktorý chceme vyriešiť?

Zhrnutie

- Problémy zo skutočného života je dobré najskôr sformulovať tak, aby bolo jasné, aké výsledky očakávame pre každý možný vstup.
- Takáto formulácia by mala byť oddelená od postupu (algoritmu) riešenia.
- Informatici merajú čas v O-čkach, ktoré abstrahujú od detailov konkrétneho počítača.
- Vytvorenie efektívneho algoritmu je umenie! Časť z toho sú finty (ako napr. dynamické programovanie).
- Pre niektoré problémy poznáme iba Nechutne Pomalé algoritmy (NP-ťažké).
- Aj napriek tomu vo veľa prípadoch vieme pomôcť.