Metódy v bioinformatike

CB #01 Informatika pre biológov

Askar Gafurov

FMFI UK

24/09/2020

Dnešné ciele

1. Pojem problému, algoritmu
2. Ukážka prevodu biologického problému na informatický
3. Efektivita algoritmu, pojem časovej zložitosti, O-notácia
4. NP-ťažké algoritmy

Formulácia problému, algoritmus

- *Formulácia problému*: jasne definované vstupné a výstupné dáta a aký výstup očakávame pre každý vstup.
- Formulácia neuvádza *akým spôsobom* sa majú zo vstupov vypočítať výstupy.
- *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 $a[1], a[2], \ldots, a[n]$ 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
    ^^^^^^^
```

Otázka na zamyslenie

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ 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
      ^^^^^^^
```

Ako túto úlohu vyriešiť?

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ 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* - počet logických operácií, ktoré program vykoná, v závislosti od množstva dát.
- Pre každú veľkosť vstupu *odhadneme najhorší možný prípad*

```
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
```

Výpočet časovej zložitosti

```
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
```

Počet operácií $+$ a $:=$

$$T(n) = \sum_{i=1}^{n} \left(\sum_{j=i}^{n} \left(1 + \sum_{u=i}^{j} 1 \right) \right) = \ldots = \frac{1}{6}n^3 - n^2 + \frac{5}{6}n$$

Zaujíma nás *najvýznamnejší* člen tejto sumy, a to je $\frac{1}{6}n^3$. Navyše, nezaujíma nás konštanta pri tom člene. Výsledok takéhoto "zjednodušenia" píšeme ako $O(n^3)$ a hovoríme, že daný algoritmus má *kubickú časovú zložitosť*.

Prečo používame O-notáciu?

Úlohou O-notácie je odpovedať na otázky typu “ak budem mať X krát viac dát, koľkokrát dlhšie budem čakať na výsledok?”

Napríklad, $T(10^5) = \frac{1}{6} \cdot 10^{15} - 10^{10} + \frac{5}{6} \cdot 10^5 = 166656666750000$

$T(2 \cdot 10^5) = \frac{1}{6} \cdot 2^3 \cdot 10^{15} - 2^2 \cdot 10^{10} + 2 \cdot \frac{5}{6} \cdot 10^5 = 1333293333500000$

$$\frac{1333293333500000}{166656666750000} = 8.000240011400564$$

Tento výpočet môžem spraviť len s najvýznamnejšími členmi:
$\frac{\frac{1}{6} \cdot 2^3 \cdot 10^{15}}{\frac{1}{6} \cdot 10^{15}} = 2^3 = 8$. Všimnite si, že na konštante $\frac{1}{6}$ nezáleží.

Prečo používame O-notáciu?

Teda, namiesto porovnávania presných funkcií stačí porovnať ich pomocou "zjednodušených" funkcií:

$$\frac{T(X \cdot n)}{T(n)} \approx \frac{(X \cdot n)^3}{n^3} = \frac{X^3 \cdot n^3}{n^3} = X^3$$

Ak by časová zložitosť bola napríklad $O(2^n)$, tak by zmena času behu algoritmu vyzerala následovne:

$$\frac{2^{X \cdot n}}{2^n} = 2^{(X-1) \cdot n}$$

Všimnite si, že pri exponenciálnej zložitosti nárast závisí nielen od X, ale aj od pôvodnej veľkosti vstupu n.

Merania

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

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$?

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

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

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

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

- Akú má časovú zložitosť výpočeť $S[i]$?

Efektívnejší algoritmus

Informatický problém

Vstup je postupnosť n kladných čísel $a[1], a[2], \ldots, a[n]$ a číslo K. Nájdite súvislý úsek tejto postupnosti $a[i], a[i+1], \ldots, a[j]$, ktorý svojim súčtom dáva číslo K.

Skúsme počítať sumy $a[i] + \ldots + a[j]$ rýchlejšie.

- Nech $S[i] = a[1] + a[2] + \ldots + a[i]$, $S[0] = 0$.
- Ak hodnoty $S[i]$ poznáme, ako vieme rýchlo zrátať súčet $a[i] + \ldots + a[j]$? $a[i] + \ldots + a[j] = S[j] - S[i-1]$
- Ako vieme spočítať hodnoty $S[i]$?

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

- Akú má časovú zložitosť výpočeť $S[i]$? $O(n)$

Problém # 2: Najkratšie spoločné nadslovo

Formulácia problému

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

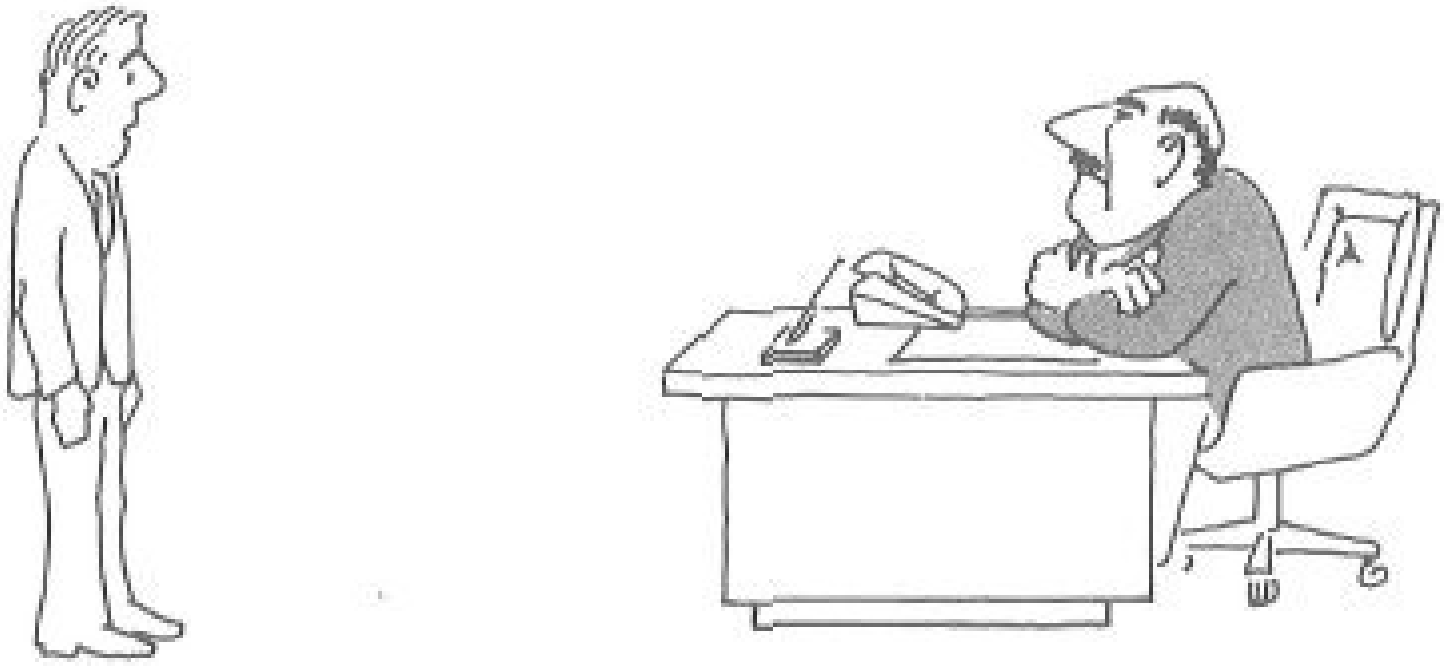
Príklad

Vstup: GCCAAC, CCTGCC, ACCTTC
Výstup: CCTGCCAACCTTC (najkratšie možné)

Problém # 2: Najkratšie spoločné nadslovo

Najlepší algoritmus?

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