Kamery

1. Časť

3/4

Popis algoritmu

Každej kamere pridelím frekvenciu rovnú počtu prvočísel, na ktoré je možné jej index rozložiť. Napríklad 12 sa rozkladá na {2, 2, 3}, čo znamená, že frekvenica kamery 12 bude 3.

O prvočíselnom rozklade môžeme rozmýšľať aj ako o prvočíslach ktorými môže byť dané číslo vydelené kým sa nestane 1. S touto informáciou môžeme spraviť algoritmus prvočíselného rozkladu. Zároveň nepotrebujeme vedieť presný číselný rozklad, stačí nám zistiť počet prvočísel v rozklade (Myslím počet prvočísel v rozklade kde exponenty nemôžu byť väčšie ako jedna, teda 12 nebude $\{2^2, 3\}$ ale {2, 2, 3})

+2

Implementácia algoritmu

Na začiatku vyvoríme premennú *i,* ktorá bude aktuálny deliteľ čísla. Nastavíme jej hodnotu 2, keďže 0 sa nedá deliť a číslo 1 je celočíselným deliteľom všetkých čísiel. Zároveň nastavím premennej *m* hodnotu *n*, pretože túto hodnotu budeme postupne deliť. Premenná p bude znamenať počet prvočísel na ktoré bolo n zatiaľ rozložené a na začiaktu bude mať hodnotu 0.

V každom cykle algoritmu číslu *m* pridelíme hodnotu m/i a zväčšíme *p* o 1 vtedy, ak je *i* celočíselným deliteľom *m.* V opačnom prípade *i* zväčšíme o 1. Tento cyklus sa bude opakovať, pokým $i<\sqrt{n}+1$ alebo pokým *i* nebude 1 (v tomto prípade je číslo *n* kompletne rozložené). Ak po skončení cyklu *n* nebude 1, zväčšíme *p* o 1, keďže aktuálne *m* bude tiež súčasťou prvočíselného rozkladu pôvodného *n*.

Na konci je teda *p* počet všetkých prvočísel v prvočíselnom rozklade alebo inak povedané, súčet exponentov všetkých rôznych prvočísel v zjednodušenom zápise rozkladu. Frekvenciu teda môžeme nastaviť na číslo *p.*

Pseudokód

```
i = 2
m = n
p = 0
opakuj pokym i < sqrt(n) + 1 a m != 1:
        ak m % i == 0:
                m /= i
                p++
        inak:
                i++
ak m != 1:
        p++
```

✓

Časová a pamäťová zložitosť algoritmu

Časová zložitosť algoritmu je $O(\sqrt{n})$. Keďže v prípade že *i* delí *m,* sa *i* neinkrementuje, v najhoršom prípade sa počas iterácie *i* neinkrementuje $\log_2(n)$-krát. To je práve vtedy ak sa *n* rozkladá na súčin v tvare 2^x. Presne teda môžeme povedať, že v najhoršom prípade prebehne približne $\sqrt{n}+\log_2(n)$ iterácii. Odmocnina z *n* je však rádovo väčšie číslo ako dvojkový logaritmus, teda zapisujeme $O(\sqrt{n})$.

Keďže nás zaujíma len počet prvočísel na ktoré bolo *n* rozložené (alebo inak povedané súčet exponentov v zjednodušenom zápise). Nemusíme si pamätať prvočísla na ktoré bolo *n* rozložené, a tak je pamäťová zložitosť $O(1)$.

Zdôvodnenie správnosti algoritmu na prvočíselný rozklad

Keďže delíme od najmenších čísel (od 2 hore), vždy sme si istý, že ak *i* delí *m*, *i* je prvočíslo. Je to kvôli tomu, že *m* nie je deliteľné číslami od 2 po *i – 1* (pretože každým týmto číslom v tomto intervale už bolo *m* delené až pokým sa dalo). Ak teda *m* nie je deliteľné číslami od 2 po i – 1, ale je deliteľné číslom *i*, znamená to, že *i* je prvočíslo.

Cyklus nám stačí opakovať len pokým je *i* menšie alebo rové $\sqrt{n}$. Je to tak, pretože *n* sa nebude nikdy rozkladať na 2 čísla z ktorých je jedno väčšie a druhé rovné alebo väčšie $\sqrt{n}$. Z toho vyplíva, že len jedno číslo v rozklade môže byť väčšie ako $\sqrt{n}$ a ak sa v rozklade nachádza ostane nám po vydelení všetkými ostatnými číslami rozkladu ktoré sú v intervale od 2 po $\sqrt{n}$, teda po konci cyklu.

Zdôvodnenie konečnosti algoritmu na prvočíselný rozklad

Cyklus algoritmu má hlavnú podmienku $i<\sqrt{n}+1$, prípad kedy sa *i* neinkrementuje je keď *i* delí *m* bez zvyšku. Keďže hodnota *m* sa vždy v tomto prípade zmení na *m / i*, tak *m* môže byť vydelené *i* maximálne $\log_2(n)$-krát. To znamená ako som už spomínal, že cyklus bude mať vždy pod $\sqrt{n}+\log_2(n)$ iterácii a teda bude vždy konečný.

Zdôvodnenie minimálneho počtu frekvencií

Všetky celočíselné delitele čísla *n* sú produkty všetkých možných neprázdnych podmnožín jeho prvočíselného rozkladu. Zároveň platí, že dve rôzne čísla, ktoré sa rozkladajú na rovnaký počet prvočísel nikdy nie sú navzájom deliteľné, pretože sa produkt akejkoľvek podmnožiny jedného čísla nikdy nerovná druhému číslu. Je to tak, pretože každé prirodzené číslo sa dá vyjadriť iba jednou množinou prvočísel a medzi dvoma takýmito množinami musí byť aspoň jedno prvočíslo rozdielne. Z toho vyplíva, že jedno číslo nikdy nebude násobkom druhého.

+1

V tejto úlohe to znamená, že takýmto číslam môžeme priradiť rovnakú frekvenciu, keďže nebudú prepojené. Z toho vyplíva, že ak sa *n* rozkladá na *m* prvočísel, stačí nám dať kamere *n* frekvenciu *m*, pretože podmnožiny rozkladu *n* môžu mať *m* rôznych veľkostí a teda jeho delitele (okrem neho samého) budú mať *m – 1* rôznych frekvencií.

Keďže každej kamere sme pridelili najmenšiu možnú frekvenciu, ktorú nemá ani jeden z jej deliteľov, budeme mať celkovo vždy minimum rôznych frekvencií.

Tento argument sám o sobě nestačí. Proč nemůže nastat situace podobná následující? Tedy že modré kamery sice nastavíme nejmenší možnou frekvencí, kterou nemá žádný z dělitelů (zelené), ale ty zelené kamery mají nechodně zvolenou frekvenci, takže nám mohl zvolit příliš velké číslo pro modrou.

2. Časť

4/6

Popis algoritmu

V tejto časti síce nevieme index kamery, ale vieme zistiť, či je má kamera index 1, keďže kamera 1 je jediná, do ktorej nevedie žiaden kábel (teda do nej vedie 0 vstupných káblov), pretože 1 je deliteľná iba sebou samou. Toto bude dôležitá informácia pre náš algoritmus.

Začatím v kamere 1, pošleme do všetkých vystupujúcich káblov hodnotu 1. Tiež si každá kamera zapamätá aktuálny čas, teda čas prvého kola. To v následujúcich kolách použijeme na zistenie aké kolo prebieha, pretože každá kamera má vnútorné hodiny.

V následúcich kolách zo všetkých kamier, ktoré majú momentálne „slovo“ (kamier ktorým prišla v predchádzajúcom kole cez nejaký vstupný kábel hodnota 1) pošleme do všetkých ich výstupných káblov hodnotu 1. Zároveň vždy nastavíme frekvenciu takých kamier na aktuálnu sekundu algoritmu. Teda na rozdiel aktuálneho a začiatočného času, ktorý sme si zapamätali. Ak z kamery nevystupujú žiadne káble, nepošleme nič. Algortimus môžeme skončiť po $\log_2(n)$ sekundách, pretože vtedy už budú mať určite všetky kamery nastavenú správnu frekvenciu.

+2

proč?

Tohle by bylo potreba víc vysvětlit.

Časová zložitosť algoritmu

Každým kolom v algoritme sa posunieme z každej kamery so „slovom“ do všetkých kamier, ktoré sú jej násobkom okrem nej samej. Keďže najmenšie násobky prirodzených čísel okrem 1, sú násobky čísla 2, najdlhšie môže algoritmus trvať $\log_2(n)$ sekúnd.] +1

Časová zložitosť výpočtou vo vnútri kamery je $O(\sqrt{n})$, pretože to žiadnej kamery nevedie viac ako $\sqrt{n}+1$ káblov, ktoré musíme skontrolovať v každom kole na to, aby sme zisitli či nejaká kamera odovzdala danej kamere „slovo“. Pamäťová zložitosť vnútorného výpočtu je O(1), pretože nám stačí zistiť či aspoň na jednom zo vstupných káblov sme dostali hodnotu 1.

Keďže nás zaujíma len komunikácia cez káble, ktorá je pomalá, stíhame všetkým kamerám nastaviť správnu frekvenciu v zadanom limite $\log_2(n)+5$ sekúnd, pretože algritmus vždy trvá maximálne $\log_2(n)$ kôl, respektíve sekúnd.

Zdôvodnenie správnosti algoritmu

V každom kole pridelíme nejakým kamerám frekvenciu podľa toho, koľkými číslami môže byť index danej kamery vydelený. Keďže sa v každej sekunde rozšíri „slovo“ z danej kamery do všetkých jej násobkov až po *n*. Vieme, že aktuálna sekunda algoritmu zodpovedá počtu čísel na „ceste“ od kamery 1 až po danú kameru, ktoré sa vždy zväčšili o nejaký svoj násobok. Teda môžeme index danej kamery vydeliť všetkými číslami v tejto ceste a dostaneme 1. ?

V predchádzajúcej časti som pridelil každej kamere frekvenciu rovnú počtu čísel v prvočíselnom rozklade indexu danej kamery. To isté sa udeje aj tu, pretože poslednýkrát dostane kamera „slovo“, keď budú násobky na ceste od 1 po index kamery tvorené prvočíslami v prvočíselnom rozklade indexu kamery. Inak povedané, najdlhšia cesta od kamery 1 po danú kameru, bude vždy tvorená prvočíslami, pretože prvočísla sa už nedajú rozložiť a tak nie je možné takúto cestu predĺžiť. ✓

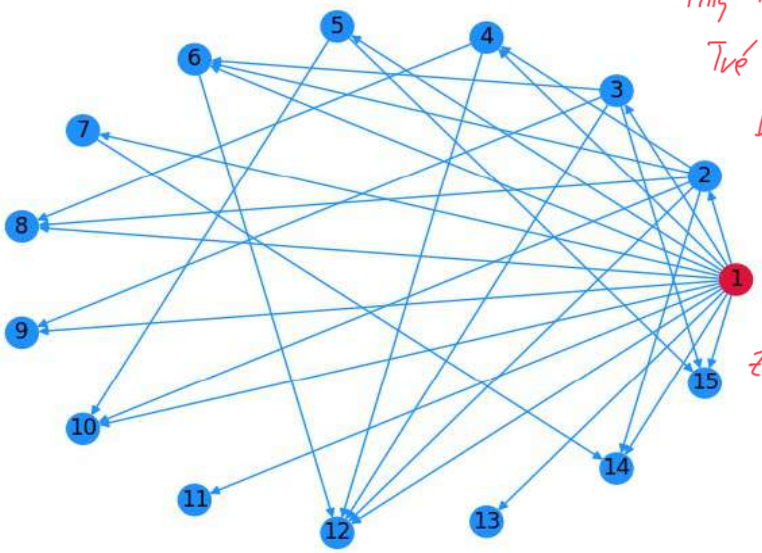
To znamená, že nastavíme každej kamere takú istú frekvenciu ako v 1. časti. Zdôvodnenie prečo má mať každá kamera frekvenciu rovnú počtu čísel v prvočíselnom rozklade jej indexu som už písal v 1. časti. +1

Zdôvodnenie konečnosti algoritmu

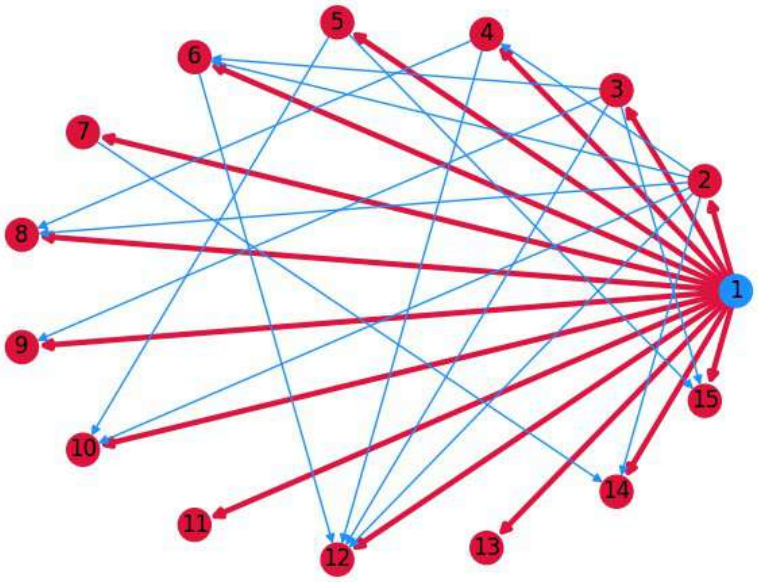
Každým kolom posunieme „slovo“ všetkým násobkom menším alebo rovným *n* všetkých kamier, ktoré majú momentálne slovo. To znamená, že vždy dôjdeme k číslam, ktoré už nemajú žiadny násobok, ktorý je menší alebo rovný *n* a teda nepošlú žiadnej kamere hodnotu 1.

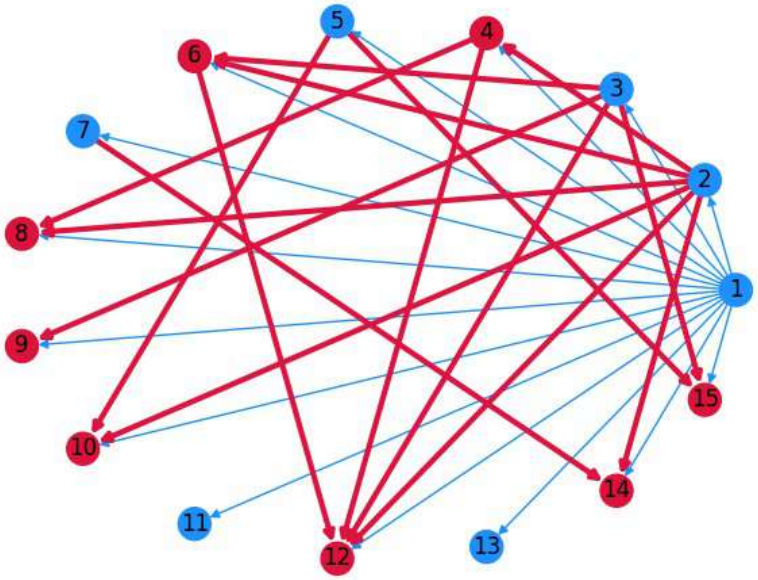
Vizualizácia algoritmu

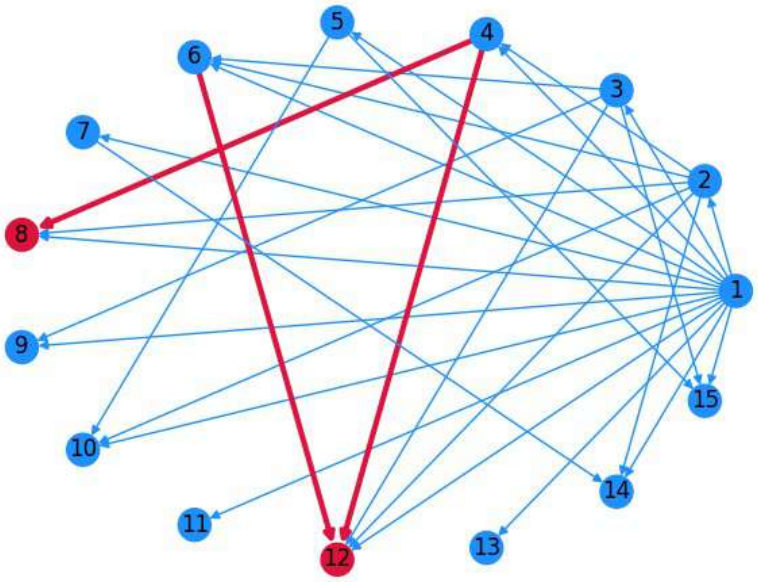
Vizualizácia algoritmu s 15 kamerami. Na červeno sú zvýraznené kamery so „slovom“, teda kamery ktoré dostali cez nejaký kábel hodnotu 1. Tieto káble sú zvýraznené tiež na červeno.



Milý Martine,
Tvé řešení je v obou částech správné. Důkazy mají drobné nedostatky – některá tvrzení by bylo potřeba trochu lépe vysvětlit. Celkem získáváš krásných 7 bodů.
Za organizátory zdraví
Martin


5
4
6
3
7
2
8
1
9
15
10
14
11
12
13


5
4
6
3
7
2
8
1
9
15
10
14
11
12
13


5
4
6
3
7
2
8
1
9
15
10
14
11
12
13