Korešpondenčný seminár z programovania
XXXII. ročník, 2014/15
Katedra základov a vyučovania informatiky FMFI UK,
Mlynská Dolina, 842 48 Bratislava

Návody k úlohám 1. kola zimnej časti kategórie T

V kategórii T neuvádzame vzorové riešenia ale skôr návody na riešenie úloh. Ich cieľom je prezradiť vám hlavnú myšlienku riešenia, aby ste podľa návodu mohli riešenie domyslieť sami. Občas teda vynecháme niektoré drobné detaily, neuvádzame implementácie dátových štruktúr a nerozpisujeme niektoré kroky.

Neuvádzame ani vzorové programy, pretože chceme, aby ste po prečítaní návodu naprogramovali tie úlohy, ktoré ste nespravili počas trvania série. Keď si tieto riešenia sami naprogramujete, naučíte sa tým oveľa viac ako pozeraním sa na zdrojové kódy.

Implementácie všeobecne známych algoritmov a dátových štruktúr môžete nájsť vo vzorových riešeniach kategórií Z a O starších ročníkov KSP alebo aj na internete.

návod písal mišof

1. Termy a výrazy

(max. 0 b za popis, 20 b za program)

V prvom rade si všimneme, že hodnota v poslednom príklade v zadaní sa nám nezmestí do 64-bitového intu. Bude teda treba buď implementovať vlastnú aritmetiku veľkých čísel (komu sa chce?) alebo použiť jazyk, ktorý ju už má (teda Python, v najhoršom BigIntegery v Jave).

Druhé užitočné pozorovanie: všetky tri funkcie zo zadania sú neklesajúce. Neformálne: čím väčší vstup, tým väčší výstup.

Pozorovanie číslo tri. Na to, aby sme zistili, či má výraz jednoznačnú hodnotu, stačí nájsť jeho najmenšiu možnú a najväčšiu možnú hodnotu a zistiť, či sa rovnajú.

V tejto chvíli zrejme mnohých pokúšalo hľadať minimum tak, že za všetky otázniky dosadíme nulu a maximum tak, že za všetky dosadíme stovku. Vďaka vyššie uvedenému pozorovaniu číslo dva by tento postup skutočne fungoval – nebyť jedného detailu. Presnejšie: keby jediné povolené binárne operácie boli sčítanie a násobenie (ktoré sú tiež na nezáporných číslach neklesajúce), skutočne by sme takto dostali najmenšiu a najväčšiu hodnotu výrazu.

My tu ale ešte máme jednu koťuhu: operátor mínus. A ten sa správa ináč. Napríklad u výrazu `?-?` je najväčšia možná hodnota 100, ale na to treba za prvý otáznik dosadiť inú hodnotu ako za druhý (100-0).

Ale na druhé zamyslenie to ani s tým mínusom nie je až také ťažké. Ak je náš výraz tvaru `U-V`, tak jeho maximum nájdeme ako maximum U mínus minimum V. A naopak, minimum výrazu `U-V` je zjavne rovné hodnote minimum U mínus maximum V.

A to už je skoro všetko. Zoberieme výraz zo zadania, rekurzívne ho delíme na menšie výrazy a tie vyhodnocujeme – teda pre každý podvýraz zistíme najmenšiu a najväčšiu hodnotu, ktorú môže nadobúdať.

Posledný detail, na ktorý si treba dať pozor: keď máme mínus, budú niektoré medzivýsledky možno záporné. Predstavte si napríklad, že máme výraz `U`, ktorého minimum je -7 a maximum 3, a výraz `V`, ktorého minimum je -5 a maximum 1. Aké je teraz maximum výrazu `U*V` a prečo? A čo minimum?

návod písal Jano

2. Transformácie tlačidiel

(max. 0 b za popis, 20 b za program)

Zistiť skutočné rozmery a uhol otočenia tlačidla je triviálne. Takže ostáva vypočítať `left` a `top` teda pozíciu ľavého horného rohu obdĺžnika pred otočením okolo stredu.

1. krok je vypočítať, ako sa musíme pohnúť z bodu (x_0, y_0) aby sme sa dostali do stredu obdĺžnika. Zoberieme vektor $(x_t, y_t) - (\frac{w}{2}, \frac{h}{2})$, vynásobíme ho s-kom a otočíme o uhol α. Vzorec na otáčanie vektora si môžete ľahko spočítať na papieri alebo ak používate C++, môžete využiť komplexné čísla. Keď bod (x_0, y_0) posunieme o tento otočený vektor, dostaneme sa do stredu obdĺžnika. Všetky uvedené premenné w, h, x_t, y_t, x_0, y_0, s a α máme uvedené na vstupe.

2. krok je posunúť sa zo stredu obdĺžnika o $\frac{ws}{2}$ vľavo a o $\frac{hs}{2}$ hore a sme v ľavom hornom rohu obdĺžnika.

Toto bolo celé riešenie. Také jednoduché! Jeho časová zložitosť je $O(1)$ na tlačidlo, dokopy o $O(n)$.

Keď implementujete geometrické úlohy, najmä tie zložitejšie, oplatí sa naprogramovať si vlastnú štruktúru bod a pretypovať si operátory + a −, aby sa vám body ľahko sčítavali a naprogramovať si funkcie na otáčanie

bodov, skalárny súčin, vektorový súčin a ďalšie, ktoré sa v geometrii často používajú. Bod a vektor samozrejme považujeme za to isté.

Najmä v zložitejších úlohách vám to zjednoduší kód a ušetríte tým mnoho času a chýb.

návod písal Jano

3. Trhanie vriec

(max. 0 b za popis, 20 b za program)

V tejto úlohe stačilo nájsť zopár dobrých patternov a naprogramovať ich. Možností je mnoho, napríklad jeden spôsob, ako sa dalo postupovať je popísaný v tomto návode.

Keby jednotlivé plochy nemuseli byť súvislé tak jednotlivé patterny (napríklad pre veľkosť 16×4) budú vyzerať takto.

```
AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBB
AAAAAAAAAAAAAAAA   ................
................   BBBBBBBBBBBBBBBB
................   ................

CCCCCCCC........   DDDD....DDDD....   EE..EE..EE..EE..   F.F.F.F.F.F.F.F.
CCCCCCCC........   DDDD....DDDD....   EE..EE..EE..EE..   F.F.F.F.F.F.F.F.
CCCCCCCC........   DDDD....DDDD....   EE..EE..EE..EE..   F.F.F.F.F.F.F.F.
CCCCCCCC........   DDDD....DDDD....   EE..EE..EE..EE..   F.F.F.F.F.F.F.F.
```

Niektoré časti plôch môžeme preklopiť a tým zlepiť niektoré pásiky dokopy bez toho, aby sme niečo pokazili.

```
AAAAAAAAAAAAAAAA   ................
AAAAAAAAAAAAAAAA   BBBBBBBBBBBBBBBB
................   BBBBBBBBBBBBBBBB
................   ................

CCCCCCCC........   ....DDDDDDDD....   EE....EEEE....EE   .FF..FF..FF..FF.
CCCCCCCC........   ....DDDDDDDD....   EE....EEEE....EE   .FF..FF..FF..FF.
CCCCCCCC........   ....DDDDDDDD....   EE....EEEE....EE   .FF..FF..FF..FF.
CCCCCCCC........   ....DDDDDDDD....   EE....EEEE....EE   .FF..FF..FF..FF.
```

Takéto zlepené pásiky sa dajú kompletne zlepiť do jedného kusa, vhodnou zmenou prvých troch riadkov. Tieto prvé riadky sú akokeby "čapaté" zatiaľ čo ostatné sú rovné.

```
AA....AAAA....AAAA....AAAA....AA     .BB..BB..BB..BB..BB..BB..BB..BB.
.AA..AA..AA..AA..AA..AA..AA..AA.     ..BBBB....BBBB....BBBB....BBBB..
..AAAA....AAAA....AAAA....AAAA..     .BB..BB..BB..BB..BB..BB..BB..BB.
.AA..AA..AA..AA..AA..AA..AA..AA.     BB....BBBB....BBBB....BBBB....BB
.AA..AA..AA..AA..AA..AA..AA..AA.     BB....BBBB....BBBB....BBBB....BB
.AA..AA..AA..AA..AA..AA..AA..AA.     BB....BBBB....BBBB....BBBB....BB
.AA..AA..AA..AA..AA..AA..AA..AA.     BB....BBBB....BBBB....BBBB....BB
.AA..AA..AA..AA..AA..AA..AA..AA.     BB....BBBB....BBBB....BBBB....BB

CCCC........CCCCCCCC........CCCC     ..DDDD....DDDD....DDDD....DDDD..
..CCCC....CCCC....CCCC....CCCC..     ....DDDDDDDD........DDDDDDDD....
....CCCCCCCC........CCCCCCCC....     ..DDDD....DDDD....DDDD....DDDD..
..CCCC....CCCC....CCCC....CCCC..     DDDD........DDDDDDDD........DDDD
..CCCC....CCCC....CCCC....CCCC..     DDDD........DDDDDDDD........DDDD
..CCCC....CCCC....CCCC....CCCC..     DDDD........DDDDDDDD........DDDD
..CCCC....CCCC....CCCC....CCCC..     DDDD........DDDDDDDD........DDDD
..CCCC....CCCC....CCCC....CCCC..     DDDD........DDDDDDDD........DDDD
```

Pokiaľ sa zaujímame len o zvislé pásiky, tak sme je všetko super. Týmto spôsobom vieme robiť súvislé plochy tak, že nájdeme políčko s každou podmnožinou. Bohužiaľ v rohoch takéto zvislé patterny nie sú kompatibilné podobnými vodorovnými. Napríklad nesledujúce dva nepasujú:

```
AA..   B...
.AA.   BB.B
..AA   .BBB
.AA.   ..B.
```

Hoci tieto by pasovali (ak chcete hlbšie pochopiť, ktoré patterny pasujú a ktoré nie, spočítajte, na koľkých políčkach je $\{\}$, $\{A\}$, $\{B\}$ a $\{A, B\}$):

```
AA..   ....
.AA.   BBBB
..AA   BBBB
.AA.   ....
```

Preto musíme patterny poohýbať, aby sa na žiadnom mieste nestretla zvislá "čapatá" vec s vodorovnou "čapatou" vecou. Inšpirujte sa nasledovnými obrázkami:

```
..A.............................    ...BBBBBBBBBBBBBBBBBBBBBBBBBBBBB
.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    B.BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
AA.AAAAAAAAAAAAAAAAAAAAAAAAAAAAA    BBB.............................
A...............................    .B..............................
A..............................A    .B............................B.
AA.AAAAAAAAAAAAAAAAAAAAAAAAAA.AA    BBB..........................BBB
.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.    B.BBBBBBBBBBBBBBBBBBBBBBBBBBBB.B
..A..........................A..    ...BBBBBBBBBBBBBBBBBBBBBBBBBB...
..A..........................A..    ...BBBBBBBBBBBBBBBBBBBBBBBBBB...
.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.    B.BBBBBBBBBBBBBBBBBBBBBBBBBBBB.B
AA.AAAAAAAAAAAAAAAAAAAAAAAAAA.AA    BBB..........................BBB
A..............................A    .B............................B.
...............................A    ..............................B.
AAAAAAAAAAAAAAAAAAAAAAAAAAAAA.AA    .............................BBB
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.    BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB.B
.............................A..    BBBBBBBBBBBBBBBBBBBBBBBBBBBBB...

.CC..CC..CC.CC....CCCC....CCCC..    DD....DDDD...DD..DD..DD..DD..DD.
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD..DD....DDDD....DDDD..
.CC..CC..CC...CCCC....CCCC....CC    DD....DDDD...DD..DD..DD..DD..DD.
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    DD....DDDD....DDDD....DDDD....DD
CC....CCCC....CCCC...CC..CC..CC.    .DD..DD..DD..DD..DD...DDDD....DD
.CC..CC..CC..CC..CC..CC..CC..CC.    ..DDDD....DDDD....DD..DDDD....DD
..CCCC....CCCC....CC.CC..CC..CC.    .DD..DD..DD..DD..DD...DDDD....DD
```

Tieto patterny nazveme *čapaté pásiky*. Všimnime si, že sú to v skutočnosti stále naše pôvodné pásiky, ktoré sme trocha počapatili aby tvorili súvislú plochu. Už je to teda súvislé aj to pasuje, ale stále tomu niečo chýba. Veru, keď máme práve dva pásiky, nedajú sa sčapatiť tak, aby boli rohy rovné. Vzniklo by niečo takéto.

```
..A..........................A..
.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.
AA.AAAAAAAAAAAAAAAAAAAAAAAAAA.AA
A..............................A
A..............................A
AA.AAAAAAAAAAAAAAAAAAAAAAAAAA.AA
.AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA.
..A..........................A..
```

Preto namiesto zvislého dvojpásikového patternu a vodorovného dvojpásikového patternu vieme použiť takúto dvojicu patternov. Nazveme ich *dvojité kríže.*

```
....MMMM....MMMM................    ................NNNN....NNNN....
....MMMM....MMMMMMMMMMMMMMMMMMMM    NNNNNNNNNNNNNNNNNNNN....NNNN....
....MMMM....MMMM................    ................NNNN....NNNN....
....MMMM....MMMMMMMMMMMMMMMMMMMM    NNNNNNNNNNNNNNNNNNNN....NNNN....
MMMMMMMMMMMMMMMMMMMM....MMMM....    ....NNNN....NNNNNNNNNNNNNNNNNNNN
................MMMM....MMMM....    ....NNNN....NNNN................
MMMMMMMMMMMMMMMMMMMM....MMMM....    ....NNNN....NNNNNNNNNNNNNNNNNNNN
................MMMM....MMMM....    ....NNNN....NNNN................
```

Ktoré navyše pasujú s patternami takéhoto typu, nazvime ich postupne patterny typu A, B, C, D.

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    ................................
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    ................................
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA    BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
................................    BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
................................    BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
................................    ................................
................................    ................................

CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
CCCCCCCCCCCCCCCC................    ........DDDDDDDDDDDDDDDD........
```

Občas však musíme použiť namiesto *dvojitých krížov jednoité kríže* alebo takéto *miešané kríže*:

```
....EEEE........   ........FFFF....   .G.G....   ....H.H.
....EEEEEEEEEEEE   FFFFFFFFFFFF....   .G.GGGGG   HHHHH.H.
....EEEE........   ........FFFF....   GGGGG.G.   .H.HHHHH
....EEEEEEEEEEEE   FFFFFFFFFFFF....   ....G.G.   .H.H....
EEEEEEEEEEEE....   ....FFFFFFFFFFFF
........EEEE....   ....FFFF........
EEEEEEEEEEEE....   ....FFFFFFFFFFFF
........EEEE....   ....FFFF........
```

Avšak tu si treba dávať pozor, že v tom smere v ktorom je kríž jednoitý, nemôžeme použiť pattern s jedným stredným pásikom. Teda kríže E, F nie sú kompatibilné s patternom typu D a kríže G, H nie sú kompatibilné s patternom typu B.

Keď rozmery plochy sú $2^r \times 2^s$, potrebujeme $r + s$ patternov. Z nich 2 budú typu A, C. Potom $2\lfloor\frac{r-2}{2}\rfloor$ bude vodorovných čapatých, $2\lfloor\frac{s-2}{2}\rfloor$ bude zvislých a potom (podľa parity r a s) bude treba doplniť 1 až 2 patterny v každom rozmere. Keď dopĺňame 1, tak kríž má byť v tom smere jednoitý, keď 2 tak použijeme dvojitý kríž a k nemu pattern typu B resp. D.

Keď už ste videli všetky tieto obrázky, mali by ste byť schopní naprogramovať riešenie bez veľkých ťažkostí. Oplatí sa spraviť to. Ak chcete, môžete si vymyslieť aj úplne iné patterny.

návod písal Jano

4. Turista programuje

(max. 0 b za popis, 20 b za program)

Na zistenie odpovede, t.j. celkový čas riešenia, počet čokolád a časy, v ktorých Turista zjedol čokolády, nám stačí vedieť, akú najdlhšiu dobu strávil riešením úlohy a koľko úloh riešil takto dlho. Z týchto dvoch údajov ľahko zrekonštruujeme odpoveď na úlohu, rozmyslite si ako. Taktiež po prečítaní tohto návodu bude ľahké ukázať, že tieto dve čísla sú jednoznačné.

Teraz si ukážeme ako tieto čísla zistiť, ak by sme vedeli, koľko presne čokolád Turista zjedol. Zistíme totiž časy riešenia všetkých úloh, len nebudeme vedieť poradie. Budeme vedieť, koľko úloh mu trvalo minútu, koľko úloh mu trvalo dve minúty, atď.

Majme niekoľko aritmetických postupností, $(i+1)$-ta vyjadruje, ako dlho budú turistovi trvať príklady po i-tej čokoláde.

$a_0 = 1, 2, 3, 4, 5, 6, 7, \ldots$
$a_1 = 1, 3, 5, 7, 9, 11, \ldots$
$a_2 = 1, 4, 7, 10, 13, \ldots$
$a_3 = 1, 5, 9, 13, \ldots$
$\ldots$

Keď vieme, že turista zje k čokolád, tak skupina optimálnych časov je n najmenších čísel (tie sa môžu opakovať) z a_0 až a_k. Napríklad 8 najmenších čísel z prvých 4 postupnosti je $1, 1, 1, 1, 2, 3, 3, 4$. Ale ako vieme nájsť najmenších n čísel z nekonečných postupností? Zjavne nebudeme nikdy brať čísla väčšie ako n, tým pádom môžme všetky postupnosti orezať aby mali len čísla do n vrátane. Potom všetky čísla jednoducho nahádžeme do (maximovej) prioritnej fronty, zahodíme všetky okrem n najmenších a potom vyberieme tie. Čísel je dokopy $n + n/2 + n/3 + n/4 + n/5 + \cdots$ čo je rádovo $n \log n$.

Ale my predsa nevieme koľko je k. To nevadí – vyskúšame všetky možnosti. Avšak nebudeme prvých n čísel z aritmetických postupností počítať vždy odznova, vieme totiž, že najmenších n čísel z $a_0 \ldots a_{k+1}$ vieme poskladať z a_{k+1} a najmenších n čísel z $a_0 \ldots a_k$.

Celý algoritmus bude vyzerať nasledovne. Nahádžeme do prioritnej fronty prvých n čísel z a_0. Postupne zvyšujeme k až po n a pri každom zvýšení spravíme toto: Hádžeme do fronty čísla z a_k, kým číslo ktoré vhadzujeme je menšie ako maximum v prioritnej fronte. Vždy, keď je v prioritnej fronte viac ako n čísel, vyhodíme maximum. Po skončení vhadzovania skontrolujeme, či sme dosiahli najlepší celkový čas a ak áno zapamätáme si dve dôležité hodnoty (najväčšie číslo v prioritnej fronte a počet toho čísla vo fronte).

Na konci len zrekonštruujeme odpoveď ktorú treba vypísať.

Pri rozumnej implementácii (prioritná fronta bude obyčajné pole, kde si na i-tej pozícii pamätáme počet prvkov i vo fronte) bude celková časová zložitosť $O(n \log n)$, pretože najviac toľko čísel dokopy spracujeme. Pamäťová zložitosť je $O(n)$.