Blok broj 2

Zadaci se mogu predati na evaluaciju na MENDO sistem
http://mendo.mk/Training.do?cid=2

Zadaci se na kraju bloka arhiviraju tako sto se spakuju samo *.c ili *.cpp datoteke. Nazivi datoteka se poklapaju sa nazivima zadataka, npr.drinks.cpp

Svi zadaci će biti ocenjeni!!!

Dva sata nakon početka blok nastave ćete dobiti korisna uputstva za rešenje PRVA TRI problema.

PIANO
http://mendo.mk/Task.do?id=598

Организатори JBOI 2015 такмичења су провели доста времена планирајући свечано затварање и они желе да свако од вас ужива те вечери. Због тога су позвали светски познатог македонског пијанисту Симона Трпчевског да наступи на свечаном затварању. Свечано затварање се одржава у великој једноспратној згради и расположиви клавири су смештени далеко од позорнице.

У целој згради, постоји тачно **N** клавира **(N <= 3)**. JBOI организатори знају локацију сваког појединачног клавира, као и коначну локацију где треба поставити клавир на позорницу. Клавир је правоугаоног облика и може се гурати у једном од четири расположива правца (лево, десно, горе и доле гледано у правоугаоној матрици зграде). Клавири се не могу ротирати ни на који начин (јер су претешки).

Симону је потребна ваша помоћ. Сви подаци са којима располажемо су приказани на правоугаоној мапи зграде. Можете ли написати програм који ће израчунати који од клавира се може догурати на позорницу у што мањем броју корака и без померања других клавира током гурања?

Улаз

Сви бројеви са којима се ради у свим задацима првог дана такмичења су цели бројеви!!!

Прва линија стандардног улаза садржи два цела броја: W и H (1 <= W<= 50, 1 <= H <= 50), висину и ширину правоугаоне матрице која представља мапу зграде.

Свака од наредних H линија ће садржати по W карактера који описују поља матрице. Ти карактери могу бити: '#' (препрека преко које се не може померити ниједан део клавира), '.' (слободно поље), '1'-'3' (обележје локације клавира), и 'F' (коначна локација за клавир позорници). Локација првог клавира је означена са једном или више цифара 1, локација другог клавира (ако он постоји) је означена са једном или више цифри 2, и локација трећег клавира (ако он постоји) је означена са једном или више цифри 3.

Клавир је правоугаоног облика величине највише 5 x 5 поља (тј. највећа ширина је 5 поља и највећа висина је 5 поља). Сви клавири на мапи имају исту ширину и висину, као и коначна локација за клавир.

Излаз

На излазу исписати два цела броја у две засебне линије: ознаку клавира који је најближи коначној локацији (тј. број 1, 2, или 3) и растојање тог клавира од коначне локације (изражено бројем потребних корака гурања). Као што је горе већ речено, кораци у којим се клавири могу гурати су лево, десно, горе, доле (гледано у правоуганој матрици). Ако постоји више клавира који су најближи коначној локацији (тј. имају једнако најкраће растојање), исписати ознаку било ког клавира.

Сматрати да ће увек бити могуће догурати бар један од клавира на коначну локацију. Могуће је да неки клавири се не могу сместити на коначну локацију, т.ј. могу бити блокирани због препрека '#' у ћелијама или због других клавира на мапи.

Пример

Улаз	Излаз
13 7 111.....	1 14

111..... ..######.... .222#..... .222#..... .....FFF. ....#....FFF.	
5 5 ..1.. .#.#2 ..#.# ....##F	1 10

Бодовање

- У тест примерима који вреде бар 30% свих поена, величина клавира ће бити 1x1 (тј. клавири ће заузети тачно једну ћелију на мапи). Уочите да други тест пример са папира задовољава ово ограничење.

РЕШЕЊЕ

Морамо да нађемо најкраћи пут од сваког клавира до крајњег одредишта . Потом тражимо најкраћи пут међу најкраћим путевима од клавира до коначне локације.
На почетку нам је дата матрица која показује где се налазе клавири, коначна локација, слободна места (-) и места неприступачна за клавир (#).

Потом креирамо граф да пронађемо најкраћи пут користећи BFS. Али проблем је што клавири могу бити различите величине, од 1x1 2x2 2x3 ,… до највише5x5. Дакле, број чворова неће бити једнак производу H(висина матрице) и W(ширина матрице), већ (H – h(висина клавира) + 1) * (W – w(ширина клавира) + 1).

Потом ћемо креирати још једну логичку или бинарну матрицу величине
[H - h + 1]*[W - w + 1].
Ова матрица ће садржати логичке вредности (boolean) и показиваће да ли постоји довољно места да се смести клавир.
Даље, пролазимо кроз матрицу и ако постоје две суседне ћелије које су 'true', у графу додајемо грану између тих чворова. Да би нашли који чворови су у питању, за сваки чвор множимо ширину бинарне матрице (W - w + 1) са бројем врста изнад врсте у којој се налази 'true' ћелија матрице и додајемо број колона које су лево од 'true' вредности.

Након што је креиран граф, позивамо BFS метод над графом са 2 параметра: број чвора где се налази један клавир и чвор крајње локације. Овај метод мора да врати најмањи број корака од полазног до циљног чвора. Да бисмо то урадили, користимо помоћни низ 'nivo' који нам памти најмањи број корака од полазног чвора до сваког чвора. У методу BFS користимо ред-queue (имплементиран као повезана листа) и када узмемо чвор из реда да би додали њему суседан чвор у ред, подесимо ниво за све суседе тог чвора (који вадимо из реда) на вредност: nivo[node] + 1. Након тпга проверимо да ли је то циљни чвор и ако јесте, вратимо његов ниво.
На крају, поредимо дужину путање од сваког клавира до крајње локације и штампамо број клавира и број корака.

Временска сложеност учитавања полазне матрице је O(W*H), сложеност налажења ширине и висине клавира је такође O(W*H). Иста сложеност важи и за обраду бинарне матрице.

Наставак обрада има у најгорем случају сложеност O(W*H). Сложеност BFS алгоритма је O(V+E) где V=W*H. Дакле, укупна сложеност је O(W*H+E).

PALM TREES
http://mendo.mk/Task.do?id=599

Влада Македоније је одлучила да засади дрворед палми дуж обале Охридског језера. Након што су палме засађене, неке од њих су порасле више него остале. Македонци не воле да сво њихово дрвеће буде исте висине (јер је то досадно), већ су задовољни уколико се дрвета разликују (то јест да нека буду висока, нека средње висине, а нека ниска).

Влада Македоније жели да анализира колико су поједини делови обале интересантни и шта ће се десити ако се одређена палма замени палмом која је виша или нижа. Приликом анализе се посебно посматрају интервали дрвореда у којима палме правилно алтернирају по висини: ако је прва палма у интервалу стого нижа од друге, тада је друга строго виша од треће, трећа стого нижа од четврте, итд. На пример, висине 3, 5, 2, 7, 6, 8 и 5 правилно алтернирају јер је $3 < 5 > 2 < 7 > 6 < 8 > 5$.

Уколико знамо висину и редни број сваке палме у дворореду, желимо да направимо програм који ће моћи да изврши више команди које следе једна за другом, при чему свака команда може бити једна од:

REPLACE *X H* – замењује палму на позицији X новом палмом висине H и команде које следе се примењују на тако измењен дрворед.

COUNT *A B* – проверава интервал дрвореда између позиција A и B (укључујући A и B) тако што израчунава најмањи број палми које је потребно уклонити да би преостале палме из интервала правилно алтернирале по висини.

Да ли знате да напишете програм који ће ефикасно извршавати такве команде?

Улаз

Прва линија улаза садржи два цела броја: N ($1 <= N <= 200\,000$), број палми у почетном дворореду и Q ($1 <= Q <= 20\,000$), број команди.

Друга линија улаза садржи N целих бројева: $T_1, T_2, \ldots, T_N$ ($1 <= T_i <= 1\,000\,000$), од којих сваки одређује висину одговарајуће палме у почетном дворореду (слева на десно).

Свака од наредних Q линија садржи по једну од две команде:

REPLACE *X H* (при чему је $1 <= X <= N$ и $1 <= H <= 1\,000\,000$)

COUNT *A B* (при чему је $1 <= A <= B <= N$).

Излаз

Програм на излазу треба да испише по једну линију за сваку команду COUNT. Свака линија излаза садржи резултат који израчунава команда COUNT, то јест најмањи број палми које је потребно уклонити. Резултати извршавања команди морају бити исписани истим оним редоследом у коме се команде појављују на улазу.

Пример

Улаз	Излаз
7 10 3 5 2 7 6 8 5 COUNT 1 2 COUNT 1 7 COUNT 4 7 REPLACE 3 8 COUNT 1 7 REPLACE 3 4 REPLACE 5 7 REPLACE 6 4 COUNT 1 7	0 0 0 2 1 0

COUNT 1 1	

Прва команда REPLACE у претходном примеру замењује трећу палму новом палмом висине 8 и након извршавања те команде висине палми у дворореду су 3, 5, 8, 7, 6, 8 и 5.
Друга команда REPLACE замењује трећу палму новом палмом висине 4 и након извршавања те команде висине палми у дворореду су 3, 5, 4, 7, 6, 8 и 5.
Трећа команда REPLACE замењује пету палму новом палмом висине 7 и након извршавања те команде висине палми у дворореду су 3, 5, 4, 7, 7, 8 и 5.
Четврта команда REPLACE замењује шесту палму новом палмом висине 4 и након извршавања те команде висине палми у дворореду су 3, 5, 4, 7, 7, 4 и 5.

Бодовање

- У тест примерима који вреде најмање 10% од укупног броја поена:
  1 <= N <= 10, 1 <= Q <= 100.
- У тест примерима који вреде додатних 40% од укупног броја поена:
  1 <= N <= 500, 1 <= Q <= 100.

RESENJE

Најпре имамо низ целих бројева.
Потом радимо обраду Q линија са упитима.
У свакој линији упит се односи на 2 цела броја a и b. Ако упит је REPLACE онда заменимо бројеве низа индексиране [a - 1] са b. Ово је једноставна и кратка операција. Ако упит је COUNT онда морамо да нађемо број целих бројева који се морају обрисати тако да подниз (почев од индекса 'a' и закључно са индексом 'b') задовољава услов. Услов је увек следећи: резултујући подниз има елементе такве да сваки члан је или већи од свог претходника и следбеника или мањи од њих. На пример, садржај може бити: 1 4 2 6 4 5.
Дакле, ако имамо COUNT упит, морамо штампати најмањи број међу целим бројевима који се бришу тако да подниз задовољи услов. Да би то решили, имамо један цео број који нам показује број целих бројева који се бришу и постављен је на '0', Потом пролазимо кроз подниз и користимо један бит тј. flag(boolean) који нам показује да ли је претходни број већи од текућег или је мањи. Потом проверимо да ли је следећи цео број исти као претходни, што значи проверу ако претходни је био мањи или проверу ако је претходни био већи (не мења се поредак). Ако јесте једнак, настављамо даље. Ако није једнак, настављамо даље, али повећамо за 1 онај бројач који смо горе поставили на '0'. И такође повећавамо тај број ако постоје два једнака броја један поред другог. Потом одштампамо тај број.

Временска сложеност најгорег случаја је O(N*Q) где N је димензија низа, Q је број упита. У најгорем случају сви упити су COUNT. Додатна оптимизација је да се не израчунава COUNT толико много пута, већ да се памти излаз COUNT упита. Али када дођемо до REPLACE упита, онда морамо ресетовати листу коју смо направили.

DRINKS
http://mendo.mk/Task.do?id=600

Није лако организовати такмичење из информатике, јер су потребни људски ресурси. То се дешава и на ЈБОИ такмичењу, где много комисија учествује у планирању и реализацији такмичења.

На JBOI 2015 такмичењу постоје две уврнуте комисије: Научна комисија и Техничка комисија. Обе комисије имају много задужења.

Данас ће ЈБОИ 2015 организатори добити N пакета (N је увек паран број) пића, и ови пакети се морају поделити у једнак број пакета између две комисије: N/2 пакета се мора дати Научној комисији и N/2 пакета се морају дати Техничкој комисији. Али, авај!!! Пакети садрже различите врсте пића и сад се око те заврзламе врти овај задатак, јер једна комисија више воли неку врсту пића. За сваки пакет нам је познат степен задовољства Научне комисије и Техничке комисије. На пример, нека имамо 4 пакета и табелу чије вредности су степени задовољства сваке комисије за сваки пакет пића.

	Пакет #1	Пакет #2	Пакет #3	Пакет #4
Научна комисија	10	10	25	30
Техничка комисија	20	30	10	5

Укупан степен задовољства једне комисије једнак је збиру степена задовољства свих пакета које је добила комисија. Можете ли помоћи JBOI 2015 организаторима и поделити N пакета између две комисије тако да разлика укупног степена задовољства прве комисије и укупног степена задовољства друге комисије је минимална? Другим речима, потребно је да покушате да поделите N пакета за две комисије, тако да свака комисија добије N/2 свих пакета, и тако да њихов укупан степен задовољства буде што мање различит.

У горе датом примеру, ако одлучимо да Научној комисији поделимо пакет#1 и пакет#2, онда ће укупан степен задовољства те комисије бити 10 + 10 = 20, и ако пакет#3 и пакет#4 поделимо Техничкој комисији, онда ће укупан степен задовољства те комисије бити једнак 10 + 5 = 15. Разлика укупног степена задовољства ове две комисије једнака је |20 - 15| = 5.

С друге стране, ако Научној комисији поделимо пакет#1 и пакет#3 (10 + 25 = 35), и ако Техничкој комисији поделимо пакет#2 и пакет#4 (30 + 5 = 35), разлика ће бити једнака |35 - 35| = 0.

Улаз

У првој линији стандардног улаза дат је један цео број N, који представља број пакета. У наредних N редова налазе се по два цела броја Ai и Bi (1 <= Ai<= 10 000 000 000 000 и 1 <= Bi <= 10 000 000 000 000), који означавају степен задовољства Научне комисије за i-ти пакет (Ai) и степен задовољства Техничке комисије за i-ти пакет (Bi).

Ограничења у вези вредности за N ћете морати пажљиво прочитати у одељку Бодовање.

Излаз

Излазни резултат мора имати три реда. У првом реду, одштампајте најмању разлику.

У другом реду, мора постојати тачно N/2 бројева који означавају које пакете треба поделити Научној комисији (тако да се одштампају у ма ком редоследу). У трећем реду постојаће тачно N/2 бројева који означавају који пакети се морају поделити Техничкој комисији (тако да се одштампају у ма ком редоследу).

Пример

Улаз	Излаз
4 10 20 10 30 25 10 30 5	0 1 3 4 2

Бодовање

- У тест примерима који вреде бар 20% свих поена важи да је 2 <= N <= 20, 1 <= Ai <= 1000 и 1 <= Bi <= 1000.

- У тест примерима који вреде додатних 40% укупног броја поена важи да је 2 <= N <= 36.

- У преосталим тест примерима N ће бити број између 40 и 100. С обзиром на релативно велике дозвољене вредности за N, такмичар треба да имплементира што бољи могући алгоритам који ће дати резултат у оквиру задатог ограничења времена извршавања програма. При томе и приближна вредност минималне разлике може бити прихваћена. Боља решења (она која проналазе мању разлику у степену задовољства две комисије) добиће више поена.

РЕШЕЊЕ

Имамо паран број N који нам говори колико следи врста и у свакој врсти имамо по 2 броја. Морамо одабрати N/2 бројева из леве колоне и исти број из десне тако да разлика међу сумама бројева које смо одбрали буде најмања

Да бисмо то урадили, можемо проћи кроз све комбинације, али је превелика сложеност тог алгоритма (O(2^n)). За неке оптимизације можемо креирати низ логичких вредности димензије N. На пример, ако имамо вредност 'true' на елемнту са индексом 5, то значи да у врсти број 5 смо узели леви број. Аналогно за случај false (што значи да је одабран елемент из десне колоне). Након тога, на случајан начин одаберемо true и false елементе (важно да их буде једнак број). Након тога можемо да нправимо трампу два елемента у итерацијама. Престајемо када итерације трају превише. Пар који учествује у трампи се бира у зависности од разлике која ће бити између суме бројева у два скупа с којим радимо с обзиром да се на mendo.mk каже да "Even an aproximate value for the minimum difference can be accepted."

ALGORITHMS
http://mendo.mk/Task.do?id=601

Македонски IOI тим чине паметни такмичари који морају што више да вежбају да би освојили медаље.

И Емил, вођа македонског тима, жели да такмичари што више вежбају. Емил је пронашао много старих такмичења из информатике, који су уредно сложени. Емил је такмичења обележио редним бројевима и формирао је листу свих такмичења. У Емиловој листи је уписан тип сваког задатка, што је заправо скраћеница за алгоритам потребан да би се задатак решио. Сва такмичења у Емиловој листи имају једнак број задатака.

Пример Емилове листе:

	Task 1	Task 2	Task3
Competition #1.	DFS	DP	BF
Competition #2.	SP	DFS	DP
Competition #3.	BF	BF	DFS
Competition #4.	SP	BF	DP

Емил планира да организује припреме тако да ће такмичарима објаснити тачно M алгоритама. За домаћи задатак, такмичари ће добити низ од неколико узастопних такмичења са Емилове листе, али тако да такмичари знају да реше бар по један задатак са сваког од тих такмичења користећи алгоритам који су учили на припремама.

Молим Вас, напишите програм који ће пронаћи најдужи низ узастопних такмичења (који задовољавају горе описане захтеве) и исписати број такмичења у том низу.

На пример, ако је M=2 и ако је Емил изабрао алгоритме {DFS, BF}, онда ће такмичари добити за домаћи задатак сва 4 такмичења са горе приказане Емилове листе, зато што свако такмичење из Емилове листе има бар један задатак типа DFS или BF.

Улаз

Три цела броја се налазе у првој линији улаза: N (1 <= N <= 100 000), који представља број такмичења, T (1 <= T <= 4), који представља број задатака на сваком такмичењу и M (1 <= M <= 3), који представља број алгоритама који се изучавају.

У свакој од наредних N линија, постоји T стрингова Aij. Сваки такав стринг је сачињен од највише 10 великих слова 'A' - 'Z'. Ови стрингови означавају алгоритме који су потребни да се реше задаци са сваког од N такмичења са Емилове листе.

Напомена: Свака два различита алгоритма су представљена са два различита стринга.

Излаз

У првој линији излаза, одштампајте број такмичења у најдужем низу. Не заборавите да Емил прави оптималан избор алгоритама, као што је описано у формулацији задатка.

Пример

Улаз	Излаз
4 3 2 DFS DP BF SP DFS DP BF BF DFS	4

SP BF DP	
7 2 1 DFS BF DFS XYZ BFS SP ABC ABC SP BFS BFS SP BFS BFS	3

Бодовање

- У тест случајевима који вреде бар 10% свих поена, важи да је 1 <= N <= 1 000 и M = 1.
- У тест случајевима који вреде додатних 10% свих поена, важи да је 1 <= N <= 100 000 и M = 1.
- У тест случајевима који вреде додатних 20% свих поена, важи да је M = 2.