#	Problem Name	Time Limit	Memory Limit
1	receipt	1 sec.	64 MB
2	algo	1 sec.	64 MB
3	power	1 sec.	64 MB
4	construct	2 sec.	64 MB
5	rbtree	1 sec.	64 MB

ამოცანა A. “ჩეკების გათამაშება”

უნიჭო გიორგიმ შეიტყო ჩეკების გათამაშების შესახებ და მალევე მეზობლად მყოფი მაღაზიის გამყიდველების ყველაზე არასასურველი კლიენტი გახდა. საქმე იმაშია, რომ ჩეკების გათამაშებაში ფულის მოგების მანიით შეპყრობილი გიორგი ყოველდღე ყიდულობს მაღაზიაში ყველაზე იაფი საღეჭი რეზინის N ცალს და ყოველ მათგანზე გამყიდველს ჩეკს ცალ-ცალკე აბეჭდინებს.

ერთხელაც გიორგის მაღაზიაში დახვდა განცხადება, რომ 1 ლარზე იაფი პროდუქციის შესყიდვა ამ მაღაზიაში აიკრძალა. უახლოეს მაღაზიამდე არც ისე ახლოა, ამიტომ უნიჭო გიორგიმ ისევ აქ გადაწყვიტა თავისი საქმიანობის გაგრძელება. იგი აპირებს, პორციებით შეიძინოს საღეჭი რეზინები ისე, რომ ყოველი პორციის ფასი არანკლებ 1 ლარის გამოვიდეს, ჯამში N საღეჭი რეზინი იყიდოს და რაც შეიძლება მეტი პორცია და შესაბამისად ჩეკი დაგროვდეს. გამოთვალეთ უნიჭო გიორგის ყოველდღიური ჩეკების მონაგარი, თუ საღეჭი რეზინის ფასი $5T$ თეთრია.

შეზღუდვები

$1 \le T \le 50$

$1 \le N \le 100$

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების receipt.in ფაილის ერთადერთი ხაზი შეიცავს ორ მთელ რიცხვს N და T.

გამომავალი ფაილის ფორმატი

გამოსატანი მონაცემების receipt.out ფაილში დაბეჭდეთ აღწერილი პროცედურის შედეგად მიღებული ჩეკების მაქსიმალური რაოდენობა. თუ გიორგი საერთოდ ვერ მოახერხებს რამის ყიდვას, დაბეჭდეთ 0.

შემომავალი ფაილის მაგალითი (receipt.in)	გამომავალი ფაილის მაგალითი (receipt.out)
8 6	2
2 1	0

განმარტება.

პირველ მაგალითში გიორგი ყოველდღე 8 ცალ 30-თეთრიან რეზინს ყიდულობს. მას შეუძლია, ცალ-ცალკე იყიდოს ოთხ-ოთხი საღეჭი რეზინი, შედეგად მიიღებს ორ ჩეკს.

მეორე მაგალითში გიორგი სულ 2 ცალი 5-თეთრიანი საღეჭი რეზინის შეძენას აპირებს და ვინაიდან მათი ჯამური ფასი 1 ლარზე ნაკლებია, იგი შენაძენს ვერ გააკეთებს.

ამოცანა B. “ალგორითმები”

ერთი არც ისე წარმატებული ყოფილი ოლიმპიელის განცხადებით, პროგრამირების შეჯიბრებზე სულ 32 ალგორითმი გამოიყენება და ვინც ყველა მათგანი იცის, უპრობლემოდ ერევა ნებისმიერ ამოცანას. განვაზოგადოთ ეს მოსაზრება. ვთქვათ, სულ შეჯიბრებზე გვხვდება A ცალი ალგორითმი. გადავნომროთ ისინი მთელი რიცხვებით 1-დან A-მდე.

როგორც იცით, სტუდენტურ შეჯიბრებში ხშირად 3-კაციანი გუნდები იღებენ მონაწილეობას და გუნდის წევრების შერჩევა ცხადია ისე უნდა მოვახდინოთ, რომ ყოველი ალგორითმისთვის გუნდში მოიძებნოს ისეთი მონაწილე, რომელმაც ეს ალგორითმი იცის.

უნივერსიტეტში გვყავს N კაცი. ჩვენთვის ცნობილია, ყოველმა მათგანმა რა ალგორითმები იცის ამ A შესაძლოდან. დავადგინოთ, რამდენი განსხვავებული გუნდის ჩამოყალიბება შეიძლება, რომელიც ნებისმიერ ამოცანას მოერევა.

შეზღუდვები

$3 \le N \le 100$

$1 \le A \le 32$

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების algo.in ფაილის პირველ სტრიქონში ჩაწერილია ორი მთელი რიცხვი N და A. შემდეგი N სტრიქონიდან თითოეული შეიცავს ერთი სტუდენტის ცოდნის აღწერას. i-ური მათგანი იწყება მთელი რიცხვით M_i - იმ ალგორითმების რაოდენობით, რომელსაც დაუფლებულია i-ური სტუდენტი. შემდეგ წერია თითო ჰარით გამოყოფილი M_i ცალი მთელი განსხვავებული რიცხვი $[1, A]$ შუალედიდან - ამ ალგორითმების ნომრები.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა algo.out ფაილში დაბეჭდეთ ერთი მთელი რიცხვი - ყველანაირი ამოცანის ამომხსნელი გუნდების რაოდენობა, რომლების შედგენა შეიძლება მოცემული სტუდენტებისგან.

შემომავალი ფაილის მაგალითი (algo.in)	გამომავალი ფაილის მაგალითი (algo.out)
4 5 2 1 4 3 2 4 5 3 1 2 3 2 1 5	3

განმარტება.

ჩვენთვის საინტერესო გუნდებია: {1, 2, 3}, {1, 3, 5}, {2, 3, 4}. {1, 2, 4} სტუდენტები რომ ავირჩიოთ, მე-3 ალგორითმზე ამოცანას ვერავინ მოერევა.

ამოცანა C. “ხარისხება”

მოცემულია ორი მთელი რიცხვი - A და B. იპოვეთ, რა მინიმალურ მთელ ხარისხში უნდა ავიყვანოთ A, რომ იგი B-ზე უნაშთოდ გაიყოს.

შეზღუდვები

$1 \leq A, B \leq 1,000,000,000,000$ (10^12)

შემომავალი ფაილის ფორმატი

შესატანი მონაცემების power.in ფაილის ერთადერთ ხაზში წერია ორი ნატურალური რიცხვი A და B.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა power.out ფაილში ჩაწერეთ ერთი მთელი რიცხვი: მინიმალური არაუარყოფითი N ისეთი, რომ A^N იყოფა B-ზე. თუ ასეთი N არ არსებობს, გამოიტანეთ -1.

შემომავალი ფაილის მაგალითი (power.in)	გამომავალი ფაილის მაგალითი (power.out)
6 4	2
10 3	-1
60 27	3

ამოცანა D. "სტრიქონის აგება"

მოცემულია სამი სტრიქონი S_1, S_2 და S_3. ჩვენ ვცდილობთ, მესამე სტრიქონი ავაგოთ პირველი ორის სიმბოლოებისგან. ამისთვის უნდა ავირჩიოთ S_1-ში ინდექსები (i1, j1) და S_2-ში ინდექსები (i2, j2) იმ პირობით, რომ i1 ≤ j1 და i2 ≤ j2, შემდეგ კი ავიღოთ პირველი სტრიქონიდან ყველა სიმბოლო i1..j1 პოზიციებზე, ხოლო მეორე სტრიქონიდან ყველა სიმბოლო i2..j2 პოზიციებზე. თუ შედეგად მიღებული სიმბოლოებით შესაძლებელია S_3 სტრიქონის აგება, ინდექსების ამ ორ წყვილს ვარგისი ვუწოდოთ.

გამოთვალეთ, მინიმუმ რისი ტოლია [j1-i1+1+j2-i2+1] მნიშვნელობა (ანუ ორივე სტრიქონიდან ამოჭრილი სიმბოლოების ჯამური რაოდენობა) მოცემული სტრიქონებისთვის ინდექსების ვარგისი შერჩევისას. ასევე გამოთვალეთ, რამდენ განსხვავებულ ვარგის შერჩევაში მიიღება ეს მინიმუმი. ორი შერჩევა ტოლია მხოლოდ მაშინ, თუ მათში ოთხივე ინდექსი მიმდევრობით ემთხვევა (მაგალითად, {i1=j1=2, i2=j2=3} და {i1=j1=3, i2=j2=2} შერჩევები განსხვავებულია).

შეზღუდვები

S_1, S_2 და S_3 სტრიქონების სიგრძეები არ აღემატება 500-ს. ისინი შეიცავენ მხოლოდ ლათინური ანბანის ზედა რეგისტრის ასოებს.

შემომავალი ფაილის ფორმატი

შესატან მონაცემთა construct.in ფაილის პირველ ხაზში წერია სტრიქონი S_1, მეორეში სტრიქონი S_2 და მესამეში სტრიქონი S_3.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა construct.out ფაილის ერთადერთ ხაზში დაბეჭდეთ ორი მთელი რიცხვი: მინიმალური ამოსაჭრელი სიმბოლოების რაოდენობა ვარგის შერჩევებს შორის და შერჩევების რაოდენობა, რომლებშიც ზუსტად ამდენი სიმბოლოა ამოსაჭრელი. თუ მოცემული ორი სტრიქონისგან მესამის აგება შეუძლებელია, დაწერეთ „0 0".

შემომავალი ფაილის მაგალითი (construct.in)	გამომავალი ფაილის მაგალითი (construct.out)
ACBB BBCAXA ABA	4 5

განმარტება.

პირველი ორი სტრიქონიდან ვერ ამოვჭრით ორ ინტერვალს ისე, რომ მათში მხოლოდ 3 ასო იყოს და შეგვხვდეს 2 'A' და 1 'B'. ჯამში 4 სიგრძის ვარგისი შერჩევები კი შემდეგია:

{i1 = 1, j1 = 1, i2 = 2, j2 = 4}. მივიღებთ {ABCA} სიმბოლოებს და მათგან "ABA" აიგება.

{i1 = 1, j1 = 3, i2 = 4, j2 = 4} და {i1 = 1, j1 = 3, i2 = 6, j2 = 6}. მივიღებთ {ACBA} სიმბოლოებს.

{i1 = 3, j1 = 3, i2 = 4, j2 = 6} და {i1 = 4, j1 = 4, i2 = 4, j2 = 6}. მივიღებთ {BAXA} სიმბოლოებს.

ამოცანა E. "წითელ-შავი ხეები"

წითელ-შავი ხე არის მონაცემთა სტრუქტურა, რომლის მეშვეობით შეიძლება ელემენტების გარკვეულ სიმრავლეში ჩამატება, ამოშლა და მოძებნა საკმაოდ სწრაფად - სიმრავლეში ელემენტების რაოდენობისგან ლოგარითმის პროპორციული ოპერაციების რაოდენობით. იგი თვითბალანსირებად ძებნის ორობით ხეს წარმოადგენს, რასაც ახერხებს წვეროების წითელ და შავ ფერებად შეღებვის და ამ ფერებთან დაკავშირებული გარკვეული ოპერაციების ჩატარების გზით.

ფორმალურად, წითელ-შავი ხე არის ფესვიანი ხე, რომლის ყოველი წვერო წითელ ან შავ ფერად არის შეღებილი და აკმაყოფილებს შემდეგ პირობებს:

1) ხის ფესვი ყოველთვის შავია.
2) წითელ წვეროს არ ჰყავს წითელი შვილი.
3) ყოველ წვეროს 0, 1 ან 2 შვილი ჰყავს.
4) ყოველ წვეროს, რომელსაც 0 ან 1 შვილი ჰყავს, გამოტოვებული შვილების ადგილზე ვაბამთ წარმოსახვით შავ ფოთლებს. ყოველი მარტივი გზა ფესვიდან წარმოსახვით ფოთლამდე შავი წვეროების ერთსადაიმავე რაოდენობას უნდა შეიცავდეს.

თქვენ მოცემული გაქვთ ორობითი ხე N წვეროთი, რომლის ფესვია წვერო 1. დაადგინეთ, რამდენნაირად შეიძლება მისი წვეროების შეღებვა ისე, რომ შედეგად წითელ-შავი ხე მივიღოთ. ვინაიდან პასუხი შეიძლება ძალიან დიდი იყოს, საკმარისია გამოთვალოთ მისი ნაშთი 1,000,000,009-ზე გაყოფისას.

შეზღუდვები

$1 \le N \le 50,000$

გარანტირებულია, რომ მოცემულია ბმული ორობითი ხე ფესვით წვეროში 1.

შემომავალი ფაილის ფორმატი

შესატან მონაცემთა rbtree.in ფაილის პირველ სტრიქონში ჩაწერილია ერთი მთელი რიცხვი N. შემდეგი N სტრიქონიდან თითოეული შეიცავს 2 რიცხვს. მათგან i-ურ სტრიქონში ჩაწერილი რიცხვები წვერო i-ს შვილების ნომრებია. თუ i-ურ წვეროს რომელიმე შვილი არ ჰყავს, მის ნაცვლად 0 ეწერება.

გამომავალი ფაილის ფორმატი

გამოსატან მონაცემთა rbtree.out ფაილში გამოიტანეთ ამოცანის პასუხი.

შემომავალი ფაილის მაგალითი (rbtree.in)	გამომავალი ფაილის მაგალითი (rbtree.out)
3 2 3 0 0 0 0	2

7 2 3 4 5 6 7 0 0 0 0 0 0 0 0	5
4 0 2 0 3 4 0 0 0	0

განმარტება.

პირველი მაგალითი. ვინაიდან ხის ფესვი აუცილებლად შავი უნდა იყოს, ასარჩევი გვაქვს მხოლოდ მისი შვილების ფერები. ამ შემთხვევაში წითელ-შავ ხეს მივიღებთ, თუ ორივე ერთი ფერის იქნება.

მეორე მაგალითი. თუ წვერო 2-ს შევღებავთ წითლად, მისი ორივე შვილი აუცილებლად შავი უნდა იყოს. ანალოგიური სიტუაციაა წვერო 3-ისთვის. ამრიგად, შეგვიძლია ან მთელი ხე შავად შევღებოთ, ან მე-2 და მე-3 წვეროებისგან ერთ-ერთი ან ორივე შევღებოთ წითლად, ან წვეროები {1, 2, 3} შევღებოთ შავად, დანარჩენები კი წითლად.

მესამე მაგალითი. როგორც არ უნდა შევღებოთ მოცემული წვეროები, ფესვიდან ფოთლებამდე (გაითვალისწინეთ წარმოსახვითი ფოთლები) გზაზე ერთსადაიმავე რაოდენობის შავი წვერო არ შეგვხვდება.