§ 7 ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММНЫЕ СИСТЕМЫ

В.И. Галочкин

ЗАДАЧИ ЗАКЛЮЧИТЕЛЬНОГО ТУРА МЕЖДУНАРОДНОЙ ИНТЕРНЕТ-ОЛИМПИАДЫ ПО ИНФОРМАТИКЕ И ПРОГРАММИРОВАНИЮ 2012 ГОДА ДЛЯ СТУДЕНТОВ ВУЗОВ РОССИИ И БЛИЖАЙШЕГО ЗАРУБЕЖЬЯ

Аннотация: *Рассмотрены решения всех девяти задач олимпиады. Тематика задач связана с построением рациональных структур данных, целочисленной арифметикой, вычислительной геометрией, расчетами на графах, выбором эвристик, поиском экстремумов. Приведен алгоритм нахождения максимальной пропускной способности ребер графа по двум непересекающимся путям. Данный алгоритм практически без изменений может быть использован для поиска двух непересекающихся путей на графе минимальной суммарной стоимости. Указан способ для определения возможности разделения без вращения плоской геометрической фигуры по разрезу в форме ломаной. В одной из задач существенно увеличена размерность исходных данных по сравнению с известной задачей. Предложен подход, предусматривающий разные способы решения в зависимости от размерности исходных данных. Другие задачи внешне похожи на известные, но требуют иного решения. Такими являются задачи о расстановке ферзей на шахматной доске и оптимальному распилу бруса.*

Ключевые слова: *Программное обеспечение, международный, олимпиада, интернет-олимпиада, студент, программирование, информатика, задача, алгоритм, решение*

Введение

Заключительный тур интернет-олимпиады проводил Поволжский государственный технологический университет (г. Йошкар-Ола) 29 мая 2012 года. В олимпиаде приняли участие 267 студентов из 69 вузов и филиалов вузов Российской Федерации, а также из Узбекистана, Таджикистана, Туркменистана, Казахстана, Кыргызстана.

Участники были отобраны по результатам отборочного тура, прошедшего ранее. Было предложено девять задач, составленных автором настоящей статьи. В соответствии с правилами командного студенческого чемпионата мира по программированию ACM (Association for Computing Machinery) для программирования предоставлялось пять часов, победители определялись в первую очередь по наибольшему количеству решенных задач, а во вторую – по меньшему времени их решения. Задача считалась решенной после прохождения всех тестов. Неудачная попытка тестирования в случае успешного решения задачи наказы-

валась 20 минутами штрафного времени. Два победителя решили по шесть задач. Каждая из задач была решена хотя бы одним участником.

Ниже приведены условия задач и описаны алгоритмы их решения. Для всех программ время работы на одном тесте не должно превышать 2 секунд, а затраты оперативной памяти 256 мегабайтов.

Задача A. Лотерея

Владелец казино для привлечения клиентов решил проводить ежедневную лотерею. В течение дня составляется список клиентов в порядке их прихода. Каждый клиент имеет общий баланс игры, выраженный положительным или отрицательным целым числом. По списку из N клиентов формируется соответствующий список балансов $A_1, A_2, \ldots, A_N$.

Игроку сообщается значение N, после чего он задает список номеров $B_1, B_2, \ldots, B_{N-1}$. Из баланса с номером $k = B_1$ вычитается следующий баланс и результат замещает баланс A_k, то есть вместо баланса A_k появляется значение $A_k - A_{k+1}$. Список балансов сдвигается на одну позицию, начиная с номера k+1. На месте k+1 оказывается значение A_{k+2}, на месте k+2 – значение A_{k+3} и т. д. При этом список сокращается на один элемент. Далее подобная операция проводится со значениями $B_2, B_3, \ldots, B_{N-1}$. В результате в списке балансов окажется единственный элемент, определяющий результат игры. Требуется по спискам балансов $A_1, A_2, \ldots, A_N$ и номеров $B_1, B_2, \ldots, B_{N-1}$ найти выигрыш или проигрыш игрока.

Ввод. В первой строке содержится значение N ($3 \le N \le 10^5$) – количество элементов в списке балансов. Во второй строке задаются N элементов списка балансов $A_1, A_2, \ldots, A_N$ ($-10^4 \le A_i \le 10^4$). В третьей строке задаются $N - 1$ номеров $B_1, B_2, \ldots, B_{N-1}$ ($1 \le B_i \le N - i$).

Вывод. В единственной строке выводится итоговое значение.

Примеры

Ввод 1	**Ввод 2**
3	3
-5 3 2	2 -6 -3
1 1	2 1
Вывод 1	**Вывод 2**
-10	5

Пояснение. В первом примере сначала производится операция (-5) – 3 = -8, а затем (-8) – 2 = -10. Во втором примере сначала производится операция (-6) – (-3) = -3, а затем 2 – (-3) = 5.

Решение. Буквальное следование правилам лотереи связано с постоянными сдвигами списка балансов. Трудоемкость такого подхода составляет $O(N^2)$. Для заданной размерности это недопустимо.

Организуем массив S с такой же индексацией, как и массив A, определяющий величину сдвига каждого элемента, и массив *Next*, задающий для каждого элемента номер следующего элемента, который должен вычитаться. Первоначально массив S заполняется нулевыми значениями, а массив *Next* значениями, на единицу превышающими индексы элементов.

На i-м шаге из баланса с номером $k = B_i$ вычитается баланс с номером *Next*[k]. Сдвиг списка балансов соответствует увеличению на 1 всех элементов массива S, начиная с номера *Next*[k]. В элемент *Next* [k] поместим значение *Next* [*Next* [k]].

Для ведения массива S организуем дерево отрезков с операцией прибавления значения на диапазоне индексов [5]. Бинарным поиском на отрезке $[B_i, N]$ найдем минимальное значение j, для которого $j - S_i = B_i$. Это значение и будет индексом нужного элемента в исходном массиве A.

Трудоемкость как бинарного поиска, так и операций с деревом отрезков составляет $O(LogN)$, поэтому общая трудоемкость алгоритма оценивается величиной $O(N\,Log^2N)$.

Задача В. Обыграть казино

Казино проводит лотерею среди клиентов по правилам задачи А. Один из неудачливых клиентов сумел сфотографировать скрытой камерой составленный за день список балансов и решил, в конце концов, обыграть казино. Каким должен быть список номеров B_1, B_2, …, B_{N-1}, чтобы достичь максимального результата игры? Найти получающийся результат.

Ввод. В первой строке содержится значение N ($3 \le N \le 10^5$) – количество элементов в списке балансов. Во второй строке задаются через пробел N элементов списка балансов A_1, A_2, … , A_N ($-10^4 \le A_i \le 10^4$).

Вывод. В первой строке выводится итоговое максимальное значение. Во второй строке указываются $N-1$ целых чисел B_1, B_2, … , B_{N-1}, определяющих последовательность операций. Если возможна разная последовательность операций, вывести любую из них.

Примеры

Ввод 1	**Ввод 2**
3	3
-5 3 2	2 -6 -3
Вывод 1	**Вывод 2**
-6	11
2 1	1 1

Пояснение. В первом примере сначала производится операция 3 – 2 = 1, а затем (-5) – 1 = -6. Во втором примере производится операция 2 – (-6) = 8, а затем 8 – (-3) = 11.

Решение. Фактически требуется расставить скобки, определяющие последовательность операций. При любом варианте расстановки скобок выражение после раскрытия скобок будет начинаться операцией $A_1 - A_2$. Поэтому значение выражения не может превышать величины $M = A_1 - A_2 + |A_3| + \ldots + |A_N|$. Покажем, что можно получить это значение.

Рассмотрим сначала случай, когда $A_i \ge 0$ при $i \ge 3$. Тогда зададим скобками следующую последовательность операций:

$(A_1-(\ldots(((A_2-A_3)-A_4)-A_5\ )- \ \ldots- A_N)) = A_1 - A_2 + A_3 + \ldots + A_N = A_1 - A_2 + |A_3| + \ldots + |A_N|$.

Пусть сейчас только $A_3 \ge 0$, а остальные A_i при $i > 3$ могут быть любыми. Обозначим через k индекс первого отрицательного элемента ($k > 3$). Выполним первую операцию $A_{k-1} - A_k = A_{k-1} + |A_k|$. Аналогично поступим с другими отрицательными элементами.

Итак, получается массив C_1, C_2,…, C_M такой, что $C_i \ge 0$ при $i \ge 3$, $C_1 = A_1$, $C_2 = A_2$, $C_3 + \ldots + C_M = |A_3| + \ldots + |A_N|$, $M \le N$, то есть задача сводится к рассмотренному в начале случаю.

Например:

5, 3, 7, -6, -2, 8, -4 →
→ 5, 3, (7 –(-6)), -2, 8, -4 →
→ 5, 3, ((7 – (-6)) – (-2)), 8, -4 →
→ 5, 3, ((7 – (-6)) – (-2)), (8 – (-4)) →
→ 5, 3, 15, 12 →5 – ((3 – 15) – 12)) = 29.

Пусть наоборот $A_i \le 0$ при $i \ge 3$. Тогда расставим скобки так:

$(\ldots((((A_1-A_2)-A_3)-A_4)-A_5) - \ \ldots- A_N) = A_1 - A_2 - A_3 - \ldots - A_N = A_1 - A_2 + |A_3| + \ldots + |A_N|$.

Снова пусть только $A_3 \le 0$, а остальные A_i при $i > 3$ могут быть любыми. Обозначим через k индекс первого положительного элемента ($k > 3$). Выполним первую операцию $A_{k-1} - A_k = A_{k-1} - |A_k| \le 0$. Аналогично поступим с другими положительными элементами.

Получим массив C_1, C_2, … , C_M такой, что $C_i \le 0$ при $i \ge 3$, $C_1 = A_1$, $C_2 = A_2$, $C_3 + \ldots + C_M = |A_3| + \ldots + |A_N|$, $M \le N$, то есть задача также сводится к рассмотренному случаю.

Например:

5, 3, -7, -6, 2, 8, -4, 9→
→ 5, 3, -7, ((-6)-2), 8, ((-4)-9)→
→ 5, 3, -7, ((-6)-2-8), ((-4)-9)→
→ 5, 3, -7, -16, -13→
→ ((((5-3) – (-7)) – (-16)) – (-13))=
= 5 – 3 + 7 +16 +13 = 38.

Все рассмотренные действия можно провести за один проход массива A.

Задача А появилась в процессе написания чекера для задачи В.

Задача С. Контрабандисты

В некоторой местности имеется сеть автомобильных дорог. Каждая дорога соединяет два разных населенных пункта. Пункты пронумерованы от 1 до *N*. Известно, какой максимальный груз можно провезти по каждой из дорог. Движение по дорогам возможно в обоих направлениях.

Контрабандисты мечтают доставить как можно больше своего нелегального товара из пункта A в пункт B. Им стало известно, что полиция готовит засаду для конфискации товара. Засада может быть организована на какой-либо дороге или в некотором населенном пункте, исключая пункты A и B. Контрабандисты решили отправить товар двумя партиями по двум разным путям, которые не имеют общих населенных пунктов и дорог. Какой наибольший груз можно гарантированно доставить из пункта A в пункт B?

Ввод. В первой строке содержатся через пробел значения *N*, *M*, *A* и *B*, где *N* – количество населенных пунктов ($3 \le N \le 30$), M – число дорог, *A* и *B* – номера начального и конечного населенных пунктов (A ≠ B). В каждой из следующих M строк находятся через пробел три значения U_i, V_i, H_i, задающие параметры очередной дороги. Значения U_i и V_i ($1 \le Ui < V_i \le 30$) определяют номера населенных пунктов на концах дороги, а величина H_i ($1 \le H_i \le 10^5$) указывает максимальный груз, который можно провезти по этой дороге. Все значения целые и положительные.

Вывод. Единственная строка должна содержать величину наибольшего гарантированного груза, который можно доставить из пункта *A* в пункт *B* с учетом пропускной способности дорог и возможности перехвата полицией одной из партий. Если двух различных путей из пункта A в пункт B не существует, вывести в выходной файл No.

Пример

Ввод 1	**Ввод 2**	**Ввод 3**
3 3 3 1	4 2 3 4	3 2 1 2
1 3 5	1 2 5	1 2 5
1 2 7	1 3 3	1 3 3
3 2 4		
Вывод 1	**Вывод 2**	**Вывод 3**
4	No	No

Решение. Требуется найти на взвешенном неориентированном графе два непересекающихся пути из начальной вершины *A* в конечную *B* так, чтобы наименьшая стоимость ребер по обоим путям была максимальна. Подобная задача о максимальном грузе для одного пути рассмотрена в [2, с. 264].

Во-первых, убедимся, что нельзя использовать «жадный» алгоритм, то есть найти лучший путь, затем удалить из графа вершины этого пути вместе с инцидентными ребрами и снова найти лучший путь. Рассмотрим граф, представленный на рис. 1:

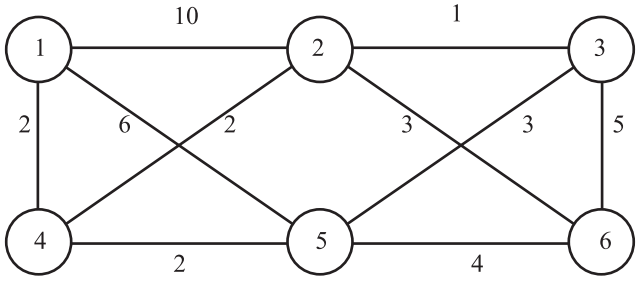

Рис. 1.
Контрпример для «жадного» алгоритма

Пусть требуется найти два пути из вершины 1 в вершину 3, для которых наименьшая стоимость дуг по обоим путям максимальна.

Лучшим путем является 1-5-6-3 с минимальной стоимостью ребер 4. Следующим лучшим путем, не пересекающимся с первым, является путь 1-2-3 с минимальной стоимостью ребер 1. Таким образом, минимальная стоимость ребер по обоим путям составляет 1. Однако пути 1-5-3 и 1-2-6-3 имеют минимальную стоимость ребер 3.

Перебор всех путей и выбор лучшей пары из них бесперспективен. Будем одновременно отслеживать два пути, продвигаясь от A к B то по одному из них, то по второму (необязательно поочередно). Для этого в процессе решения будем неявно строить новый граф, вершинами которого являются пары вершин исходного графа. В каждой паре, кроме пар из двух начальных и двух конечных вершин, окажутся разные вершины, так как пути не должны пересекаться. Для определенности будем считать, что первой в паре располагается вершина с меньшим номером.

Ребро из некоторой первой пары вершин в какую-либо вторую пару при поиске двух путей из A в B описывается следующими правилами:

- в исходном графе есть ребро L, соединяющее одну из вершин первой пары с одной из вершин второй пары;
- две другие вершины из этих пар совпадают;
- два пути, восстановленные в исходном графе из второй пары в обратном направлении к паре вершин (A, A), где A – начальная вершина, не пересекаются;
- стоимостью ребра между парами вершин является стоимость ребра L.

В приведенном примере ребро (1, 1) – (1, 4) имеет стоимость 2, а ребро (2, 5) – (5, 6) – стоимость 3.

Задача сводится к нахождению пути в новом графе из вершины (A, A) в вершину (B, B), для которого минимальная стоимость ребер максимальна. Поскольку стоимости ребер положительны, то удобно модифицировать алгоритм Дейкстры (поиск в ширину кратчайшего пути). В новом графе не более N^2 вершин, поэтому общая трудоемкость алгоритма $O(N^4)$, что вполне допустимо для заданной в условии размерности графа.

В нашем примере таким путем будет (1, 1) – (1, 2) – (2, 5) – (5, 6) – (3, 5) – (3, 3), у которого минимальная стоимость ребер 3. По этому пути восстанавливаются два пути в исходном графе 1–5–3 и 1–2–6–3. У каждого из них минимальная стоимость ребер 3.

Есть еще один нюанс. Если имеется ребро $A – B$ большой стоимости, то недостаточно проверять отсутствие пересечения двух путей по промежуточным вершинам. Нужно отдельно отсечь случай нахождения двух одинаковых путей $A – B$. В примере 1 из условия приведен такой случай.

Заметим, наконец, что граф по парам вершин не нужно явно размещать в памяти в виде матрицы стоимостей или какой-либо другой структуры. Описанный алгоритм практически без изменений может быть использован для поиска двух непересекающихся путей на графе минимальной суммарной стоимости.

Задача D. Клад

Кладоискатели обнаружили люк в подземелье с сокровищами, закрытый неподъемной квадратной чугунной плитой. К счастью, плита имеет сквозную трещину, разделяющую ее на две части, которые не лежат полностью одна внутри другой. Трещина представляет собой ломаную без самопересечений и самокасаний. В распоряжении кладоискателей имеется лебедка. Мощности лебедки хватает на то, чтобы двигать в определенном направлении без вращения каждую из частей, не поднимая ее, но не всю плиту целиком. Требуется по форме трещины определить, могут ли кладоискатели освободить люк.

Ввод. В первой строке находится число количество тестовых блоков L ($1 \le L \le 10$). В первой строке каждого тестового блока указывается N ($3 \le N \le 100$)

– количество вершин ломаной для очередного варианта теста. Следующие N строк содержат через пробел пары целых чисел (X_i, Y_i) – координаты вершин ломаной ($-1000 \le X_i, Y_i \le 1000$). Ломаная получается путем последовательного соединения точек в данном порядке. Направление обхода вершин ломаной может быть произвольным. Точки (X_1, Y_1) и (X_N, Y_N) лежат на одной или разных сторонах квадрата, другие точки ломаной – внутри квадрата.

Вывод. Выводится L строк со значением Yes или No, определяющих возможность либо невозможность разъединения квадрата путем перемещения его частей по плоскости без вращений для соответствующего варианта.

Пример

Ввод 1	**Ввод 2**
1	1
5	4
0 0	3 0
0 1	4 1
1 1	1 1
2 2	2 0
2 0	
Вывод 1	**Вывод 2**
Yes	No

Решение. Обозначим через V_1 (X_1, Y_1) и V_N (X_N, Y_N) точки ломаной, лежащие на стороне (сторонах) квадрата. Рассмотрим параллельный перенос в направлении вектора E одной из частей квадрата. Из какой-либо вершины ломаной V_i проведем прямую, параллельную вектору E. Если эта прямая пересекает прямую V_1V_N в точке W_i, то в случае возможности разъединения квадрата отрезок V_iW_i не должен содержать внутренних точек другой части квадрата. Это равносильно тому, что при обходе ломаной в порядке вершин от V_1 до V_N все параллельные вектору E прямые V_iW_i расположены так, что прямая V_iW_i не может быть ближе к точке V_1, чем прямая V_jW_j при $i > j$. На рисунке показан случай, когда условие нарушено (прямая $V_{N-1}W_{N-1}$ ближе к точке V_1, чем прямая $V_{N-2}W_{N-2}$).

Допустим, что разъединение в направлении вектора E возможно. Насколько можно изменить это направление? Предельное возможное изменение приведет к тому, что для некоторой вершины V_i вектор E окажется направлен вдоль звена ломаной V_iV_{i+1}, то есть отрезки V_iW_i и $V_{i+1}W_{i+1}$ окажутся на одной прямой. Пересечения некоторого звена, как показано на рис. 2, быть не должно.

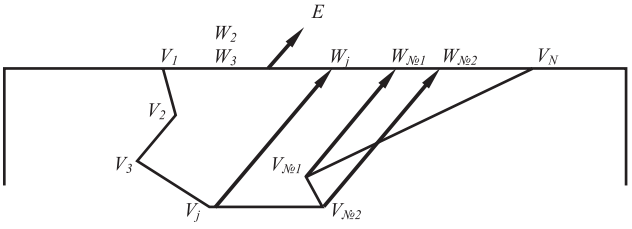

Рис. 2.
Пример неудачного разъединения в направлении вектора E

Из всех этих соображений строится следующий алгоритм. Во внешнем цикле по i ломаная обходится от вершины V_1 до V_{N-1}. При каждом значении i за вектор E принимается отрезок V_iV_{i+1}. Далее во внутреннем цикле по j находятся уравнения прямых, проходящих через вершины V_j и параллельные вектору E в форме с угловым коэффициентом $Y = KX + B_j$. Если при некотором значении i последовательность величин B_j монотонна, то части квадрата можно разъединить. Нужно отдельно учесть случай, когда прямые параллельны оси Y, то есть имеют уравнения $X = B_j$. И в этом случае нужно проверять монотонность последовательности величин B_j. Трудоемкость алгоритма составляет $O(N^2)$ для одного варианта из тестового блока.

Задача E. Ферзи

Имеется шахматная доска $N \times N (4 \le N \le 40)$, на которой по горизонталям слева направо записаны в порядке возрастания целые числа от 1 до N^2. Выбирается такой набор из N клеток, что любые два ферзя, стоящие на этих клетках, не бьют друг друга. Иными словами, любые две клетки набора находятся на разных горизонталях, вертикалях и диагоналях. Найти максимальную сумму чисел в таких наборах.

Ввод. В единственной строке задается значение N.

Вывод. В единственной строке выводится максимальное значение суммы.

Пример

Ввод

4

Вывод

34

Решение. Классическая задача о расстановке восьми ферзей, которые не бьют друг друга [3, с. 193], может быть обобщена на случай большей доски, но имеет высокую трудоемкость. Число расстановок определено до N=26 [4]. Первая расстановка найдена до N=45 [6]. В нашем случае подобный подход невозможен.

Докажем, что сумма чисел по любому набору клеток, попарно находящихся на разных горизонталях и вертикалях, постоянна. Иными словами, применительно к шахматам речь идет о ладьях, а не о ферзях.

Возьмем произвольный набор. Обозначим координаты парами вида (A, B), где A – номер горизонтали, а B – номер вертикали. В наборе должна быть ровно одна клетка из каждой горизонтали и вертикали, то есть числа набора представляются парами координат (1, I_1), (2, I_2),…, (N, I_N), где $I_1, I_2, \dots, I_N$ различны и принимают значения от 1 до N. Паре (k, I_k) соответствует число $N \times (k\text{-}1) + I_k$. Тогда по формулам арифметической прогрессии сумма этих чисел равна

$$S = \sum_{k=1}^{N} (N \times (k-1) + I_k) = N \times \sum_{k=1}^{N} (k-1) + \sum_{k=1}^{N} I_k = N \times (N^2 + 1)/2$$

Остается подставить N в полученную формулу.

Задача F. Малыш и Карлсон

Обеденный перерыв Карлсона составляет N микросекунд. Меню состоит из трех видов бутербродов: с ветчиной, сыром и вареньем, поглощаемых за C_1, C_2 и C_3 микросекунд соответственно. Малыш мечтает, чтобы его друг Карлсон съел как можно больше бутербродов. При одинаковом количестве бутербродов наилучшим становится вариант, при котором от перерыва останется как можно меньше времени. Малыш понимает, что в оставшееся свободное время шалости Карлсона могут иметь непредсказуемые последствия. Помогите Малышу правильно рассчитать количество бутербродов каждого вида.

Ввод. В первой строке задается значение N ($1 \le N \le 2 \times 10^9$). Во второй строке содержатся C_1, C_2 и C_3 ($1 \le C_1, C_2, C_3 \le 2 \times 10^9$). Все числа целые.

Вывод. В первой строке вывести общее количество съеденных бутербродов. Во второй строке указать время из обеденного перерыва, свободное от еды. В третьей строке вывести количество съеденных бутербродов каждого вида соответственно. Если решений несколько, вывести любое из них.

Пример

Ввод

14

5 3 7

Вывод

4

0

1 3 0

Решение. Без ограничения общности будем считать, что $C_1 \le C_2 \le C_3$. Обозначим $C = \min(C_1, C_2, C_3)$, $D_i = N \bmod C_i$, $D = N \bmod C$.

Очевидно, что можно максимально съесть $M = [N/C]$ бутербродов, где квадратными скобками обозначена целая часть числа. Если при этом будут съедены наименее «трудоемкие» бутерброды, то останется $D = N - MC$ микросекунд ($0 \le D < 2\times10^9$). Рассмотрим два случая.

$C \le 10^6$. Тогда $D < C \le 10^6$. Для минимизации этого остатка нужно вместо части менее «трудоемких» бутербродов съесть более «трудоемкие».

Обозначим $A = C_2 - C_1$ и $B = C_3 - C_1$. Если съесть X бутербродов за C_2 микросекунд каждый и Y бутербродов за C_3 микросекунд, то добавочно будет затрачено $AX + BY$ микросекунд. Задача сводится к поиску значений X и Y таких, чтобы выполнялись условия

$AX + BY \le D$,

$D - (AX + BY) \to \min$

Если $A = 0$ и $B = 0$, то $X = 0$ и $Y = 0$. Если $A = 0$, но $B \ne 0$, то $X = 0$, а $Y = [D/B]$. Если, наконец, $A \ne 0$, то различных значений X может быть $[D/A]$, то есть не более 10^6.

В цикле для каждого значения X определим $Y = D - [(AX) / B]$. Остается сохранить лучшие значения X и Y. Тогда потребуется съесть $M - (X + Y)$, X и Y бутербродов соответственно.

$C > 10^6$. Тогда $C_1, C_2, C_3 > 10^6$. Задача сводится к поиску значений U, V и W, определяющих количество бутербродов каждого вида, таких, чтобы совместно выполнялись условия

$C_1U + C_2V + C_3W \le N$,

$U + V + W = M$,

$N - (C_1U + C_2V + C_3W) \to \min$

Выполним полный перебор. Можно съесть не более $[N/C_1]$ бутербродов первого вида. Поскольку $C_1 > 10^6$, то $U < 2\times10^3$. В двойном цикле для каждого значения U переберем $N - [(C_1U) / C_2]$ значений V, а для каждой пары значений U и V однозначно определим число бутербродов третьего вида $W = D - [(C_1U + C_2V) / C_3]$. Максимальное число шагов не более 4×10^6.

Подобная задача приведена в [1, с. 69], но там речь идет о двух бутербродах, а значения параметров на три порядка ниже.

Задача G. Автостанция

В некотором районе планируется построить на автомобильной дороге новую автостанцию. Дорога представляет собой отрезок прямой, соединяющий пункты A и B. На карте района известны координаты этих пунктов (u_1, v_1) и (u_2, v_2) соответственно. От станции будут проложены новые прямолинейные дороги к N населенным пунктам с координатами (x_i, y_i). Известно, что затраты на проведение каждой дороги равны в некоторых единицах квадрату длины дороги. Некоторые населенные пункты могут находиться на отрезке AB или его продолжении. Тем не менее, чтобы не ограничивать движения на дороге AB, новая дорога все равно строится. Требуется указать такие координаты автостанции, чтобы общие затраты на строительство всех дорог были минимальны.

Ввод. В первой строке задается значение N ($1 \le N \le 20$). Во второй строке указываются u_1, v_1, u_2, v_2 ($-100 \le u_i, v_i \le 100$). В следующих N строках задаются координаты x_i, y_i ($-100 \le x_i, y_i \le 100$) населенных пунктов. Все числа целые. Разные населенные пункты имеют разные координаты.

Вывод. Вывести с 10 знаками после запятой через пробел два действительных числа – координаты автостанции.

Пример

Ввод

2
0 0 4 4
1 1
3 3

Вывод

2.0000000000 2.0000000000

Решение. Пусть (x, y) – неизвестные координаты станции. Общие затраты выражаются функцией

$P = (x-x_1)^2 + (y-y_1)^2 + (x-x_2)^2 + (y-y_2)^2 + \ldots + (x-x_n)^2 + (y-y_n)^2$

После подстановки y из уравнения прямой $y = kx + b$, проходящей через точки A и B, получим квадратный трехчлен. Остается найти его минимум на отрезке $[u_1, u_2]$. В случае $u_1 = u_2 = u$ подставляется $x = u$ и находится минимум трехчлена P по y на отрезке $[v_1, v_2]$.

Задача H. Распил бруса

На пилораму привезли брус длиной L метров. Требуется сделать N распилов. Распилы делят брус на части, длина которых выражается натуральными числами. Стоимость одного распила равна длине распиливаемого бруса. Определить минимальную стоимость распила.

Ввод. В первой строке содержатся натуральные числа L ($2 \le L \le 2\times10^5$) – длина бруса и N ($N < L$) число распилов.

Вывод. В единственной строке вывести минимальную стоимость распилов.

Примеры

Ввод 1	**Ввод 2**
3 2	5 3
Вывод 1	**Вывод 2**
5	10

Решение. Задача внешне похожа на известную [3, с. 289]. Для небольших размерностей она решается с помощью подходов динамического программирования. Если $F(L, K)$ – минимальная стоимость K разрезов бруса длины L, то $F(L, 1) = L$. Справедливо рекуррентное соотношение

$$F(L,K) = L + \min(F(L-S,M) + F(S,K-M-1)),$$

где минимум берется по всем значениям S от 1 до L-1 и M с учетом того, что брус длины D можно разрезать не более D-1 раз. При заданной размерности такой способ решения не годится.

Если $L = N + 1$ проходит решение, когда начальный отрезок делится пополам (с точностью до 1). Затем каждый из полученных отрезков, если его длина не менее 2, делится так же, и процесс продолжается до конца.

Пусть сейчас $L > N + 1$. Разумным кажется первым разрезом отделить отрезок длины N и далее работать с ним. Это не так. При $L = 9$ и $N = 7$ такой подход дает стоимость разрезов 29. Действительно, 9→7+2, 7→4+3, 4→2+2, 2→1+1, 2→1+1, 3→2+1, 2→1+1. Однако расчеты дают минимальную стоимость 27: 9→5+4, 4→2+2, 2→1+1, 2→1+1, 5→3+2, 2→1+1, 3→2+1.

Если в этом примере раньше делить отрезок длины 2, остающийся после первого разреза, то также получится общая стоимость 27. Применим другую эвристику. Переберем точки первого разреза. При каждом варианте далее будем делить пополам (с точностью до 1) минимальный отрезок. Однако при $L = 10$ и $N = 13$ получается стоимость 45, вместо минимальной 44.

Следующее решение предложил И. Карлин. Переформулируем задачу. Будем наоборот собирать брус длины L из N+1 бруска меньшей длины. Иными словами, число L делится на N+1 натуральных слагаемых. Далее два каких-либо слагаемых заменяется их суммой. Так продолжается, пока не остается единственное значение, которое и требуется минимизировать.

Этот процесс можно представить бинарным деревом, в листьях которого находятся N+1 чисел, в сумме составляющих L. Они входят в общую стоимость при каждой операции сложения.

Сделаем несколько разумных предположений:

- Наиболее часто прибавляемые числа должны быть минимальными.
- Наименьшее слагаемое должно быть равным 1.
- Выгодно придерживаться «жадной» стратегии, то есть на каждом шаге складывать два самых маленьких числа.

В результате приходим к следующему решению. Присвоим N слагаемым значения 1, а последнему слагаемому – значение L-N. На каждом шаге будем складывать два наименьших слагаемых, заменяя их суммой, пока не останется одно значение.

Была проведена проверка правильности такого подхода путем сравнения с расчетами по рекуррентным формулам. Получено полное совпадение результатов при $2 \le L \le 500$ и $1 \le N < L$. Пример для $L = 10$ и $N = 7$, дающий общую стоимость 29 единиц, показан на рис. 3.

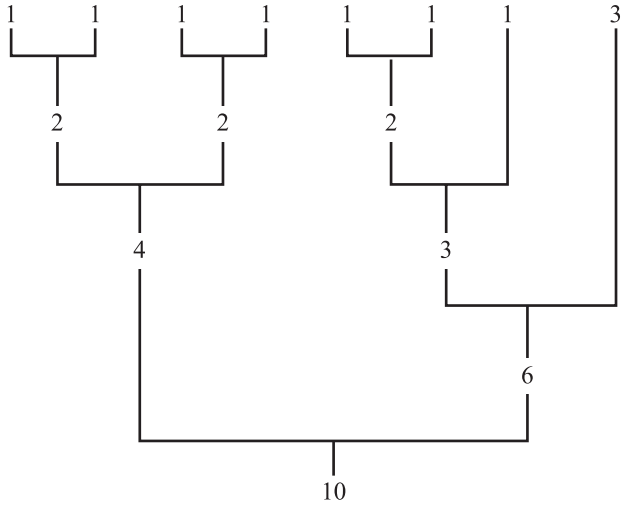

Рис. 3. Дерево распила бруса для $L = 10$ и $N = 7$

Поиск двух минимальных слагаемых при максимальной размерности не проходит по времени. Можно кардинально уменьшить как трудоемкость алгоритма, так и объем необходимой памяти.

Итак, имеется N элементов со значением 1, соответствующих длинам минимальных отрезков, и один элемент со значением L-N. Не нужно хранить их в массиве. Из них составляем $N_2 = [N/2]$ элементов со значением 2, которые также не стоит сохранять. Если N нечетно, одна единица остается. Следующую пару могут составить два минимальных элемента со значениями 1, 2, L-N. В результате сложения может появиться элемент еще одного типа.

Далее перейдем к элементам со значением 2. Если меньших элементов нет, соберем и сложим все пары элементов со значением 2. Выбор следующей пары будет производиться не более чем из четырех различных элементов.

Затем переходим к элементам со значениями 4, 8, 16 и т. д. Прежде чем рассматривать элементы со значением 2^k, нужно произвести сложение оставшихся элементов предыдущего уровня, которые имеют значения, не превышающие 2^k.

Таким образом, поиск в цикле двух минимальных элементов для сложения производится только среди четырех элементов. Трудоемкость алгоритма оценивается величиной $O(LogN)$. Поскольку нет необходимости сохранять значения элементов, объем необходимой памяти составляет $O(1)$.

Задача I. Кометы

Астроном Коперни увлечен кометами. Он определил новую физическую единицу яркости Co и вывел закон нахождения яркости комет в зависимости от времени в виде $Y=At+B$. В дальнейших исследованиях ему удалось рассчитать коэффициенты A и B для всех известных комет. Коперни в любой момент времени всегда наблюдает в свой телескоп комету с наибольшей яркостью. Как только у другой кометы яркость оказывается большей, он переносит на нее свои наблюдения. Для заказа нового телескопа ему необходимо знать минимальную яркость наблюдаемой им кометы в определенный период времени. Помогите астроному Коперни.

Ввод. В первой строке указаны N и T ($1 \le N, T \le 100$) – число комет и период наблюде-

ния. В следующих N строках даны коэффициенты A_i и B_i ($-100 \le A_i, B_i \le 100$) для всех N комет. Все числа N, T, A_i, B_i целые. Необходимо иметь в виду, что яркость кометы в единицах Co может быть как положительной, так и отрицательной.

Вывод. В единственной строке вывести действительное число с точностью 10 знаков после запятой – минимальную яркость кометы, которую будет наблюдать астрономом в период от 0 до T.

Пример

Ввод

2 4
1 1
-1 5

Вывод

3.0000000000

Решение. Функция яркости наблюдаемой кометы имеет вид $Z(t) = Max\,(A_i t + B_i)$, где максимум определяется для $1 \le i \le N$. Требуется найти $Min\ Z(t)$ на отрезке $[0, T]$. Легко доказать, что $Z(t)$ имеет единственное минимальное значение. Если существует j, для которого $A_j = 0$, минимум может достигаться на некотором отрезке. Для решения задачи наиболее рационально использовать троичный поиск [7] по значениям функции $Z(t)$ на отрезке $[0, T]$.

Отметим в заключение, что наиболее распространенная ошибка участников олимпиады заключалась в недооценке трудоемкости вычислений, в результате чего тесты максимальной размерности не проходили по времени. Победители олимпиады решили по шесть задач. Каждая задача была решена хотя бы одним участником.

Библиография:

1. Меньшиков, Ф. В. Олимпиадные задачи по программированию / Ф. В. Меньшиков. – СПб.: Питер, 2003. – 315 с.
2. Порублев, И. Н. Алгоритмы и программы. Решение олимпиадных задач / И. Н. Порублев, А. Б. Ставровский. – М.: Изд. дом «Вильямс», 2007. –480 с.
3. Скиена, С. С. Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям / С. С. Скиена, М. А. Ревилла. – М.: КУДИЦ-ОБРАЗ, 2005. – 416 с.
4. http://24forums.ru/forum/topic_768.
5. http://e-maxx.ru/algo/segment_tree.
6. http://queens.cspea.co.uk/csp-q-1stsols.html.
7. http://ru.wikipedia.org/wiki/Двоичный поиск.

References (transliteration):

1. Men'shikov, F. V. Olimpiadnye zadachi po programmirovaniyu / F. V. Men'shikov. – SPb.: Piter, 2003. – 315 s.
2. Porublev, I. N. Algoritmy i programmy. Reshenie olimpiadnykh zadach / I. N. Porublev, A. B. Stavrovskiy. – M.: Izd. dom «Vil'yams», 2007. –480 s.
3. Skiena, S. S. Olimpiadnye zadachi po programmirovaniyu. Rukovodstvo po podgotovke k sorevnovaniyam / S. S. Skiena, M. A. Revilla. – M.: KUDITs-OBRAZ, 2005. – 416 s.
4. http://24forums.ru/forum/topic_768.
5. http://e-maxx.ru/algo/segment_tree.
6. http://queens.cspea.co.uk/csp-q-1stsols.html.
7. http://ru.wikipedia.org/wiki/Dvoichnyy poisk.