Разбор задачи Манхэттен

В данной задаче необходимо было сложить разницу между соответствующими координатами, а именно реализовать формулу $|x_1 - x_2| + |y_1 - y_2|$. Пример решения на PascalABC:

```
var x1, x2, y1, y2: integer;

begin
  read (x1, y1, x2, y2);
  writeln (abs(x1 - x2) + abs(y1 - y2));
end.
```

Замечение. Для того, чтобы сдать задачу на Free Pascal, необходимо в самом начале кода прописать строку {$MODE OBJFPC}, чтобы тип данных integer был 32-битным. Также эта строчка нужна в Free Pascal, чтобы объявлять большие массивы.

Разбор задачи Пирамида

Заметим, что всегда выгодно строить пирамиду, каждый слой которой (кроме нижнего) отличается ровно на единицу, а наверху ровно 1 кубик. Пусть слоев k, тогда общее число кубиков равно $\frac{k \cdot (k+1)}{2}$. Так как общее число кубиков не должно превышать n, то необходимо найти наибольшее такое k, что $\frac{k \cdot (k+1)}{2} \leq n$. Это можно сделать бинарным поиском по k. Другой вариант решения — найти корни квадратного уравнения $\frac{k \cdot (k+1)}{2} = n$, округлить их вниз при надобности и среди них выбрать наибольший.

Разбор задачи Олимпиада

Количество способов выбрать из n задач k задач можно высчитать через биномиальные коэффициенты по следующей формуле: $\frac{n!}{k! \cdot (n-k)!}$. В данной задаче надо было реализовать следующую формулу:

$$\frac{a!}{ka! \cdot (a - ka)!} \cdot \frac{b!}{kb! \cdot (b - kb)!} \cdot \frac{c!}{kc! \cdot (c - kc)!}$$

Обратите внимание, что простая реализация не будет работать в языках со строгой типизацией, так как 30! очень большое число и не влезает в стандартные типы данных, поэтому необходимо сократить $n!$ на $k!$ и получить формулу такую: $\frac{n \cdot (n-1) \cdot \ldots \cdot (k+1)}{(n-k)!}$, где не происходит переполнение в беззнаковых 64-битных типах данных.

Разбор задачи Монеты

Решение на 20-60 баллов

Переберем за три вложенных цикла коэффициенты p_a, p_b, p_c от 1 до n и проверим, что $p_a \cdot a + p_b \cdot b + p_c \cdot c = n$ и среди всех таких выберем набор с минимально суммой $p_a + p_b + p_c$. Заметим, что коэффициент p_c можно было не перебирать третьим циклом, а вычислить формулой $p_c = \frac{n - p_a \cdot a - p_b \cdot b}{c}$ на основе зафиксированных p_a, p_b.

Решение на 90 баллов

Заметим, что коэффициент p_a не может быть больше или равен b, потому что иначе можно каждые b монет достоинства a заменить на a монет достоинства b, что не ухудшает ответ на задачу (общее кол-во монет), так как $a < b$. Поэтому можно перебирать коэффициенты не до n, а до b и c.

Решение на 100 баллов

Переберем коэффициент $p_a \leq b$, для вычисления коэффициентов p_b и p_c с минимальной суммой, таких, что $n - p_a \cdot a = p_b \cdot b + p_c \cdot c$ воспользуемся стандартным алгоритмом. Решим диофантово уравнение и найдём некоторое решение p_b и p_c. Заметим, что к p_b можно прибавлять (отнимать) $k \cdot \frac{c}{gcd(b,c)}$, а к p_c можно отнимать (прибавлять) $k \cdot \frac{b}{gcd(b,c)}$ до тех пор, пока $p_a, p_b \geq 0$, где $gcd(b, c)$ — наибольший общий делитель b, c, а k — любое целое число. При выборе как можно меньшего значения k мы получим решение с наименьшей суммой $p_a + p_b$.

Разбор задачи Секрет

Формальная модель

Дан неориентированный граф, в нем зафиксированы две вершины a, b. Необходимо найти все вершины, удаление которых приводит к тому, что вершины a, b будут в разных компонентах связности.

Алгоритм решения на 90

Переберем вершину v, удалим вершину v, проверим, что вершины a и b всё ещё в одной компоненте связности, например, используя обход в глубину.

Алгоритм решения на 100

Запустим обход в глубину из вершины a. Для каждой вершины v посчитаем время входа $tin[v]$ и минимальное время вершины, которая достижима из поддерева обхода в глубину вершины v по обратным рёбрам — $up[v]$. Построим простой путь из a в b, состоящий только из прямых ребёр обхода. Обозначим последовательные вершины пути $path_1 = a, path_2, ..., path_k = b$. Вершина $path_i, 1 < i < k$ входит в ответ тогда и только тогда, когда выполняется $tin[path_i] \leq up[path_{i+1}]$.

Доказательство

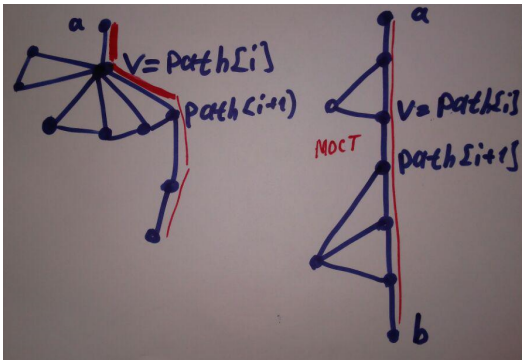

Рис. 1: Достаточность.

- Достаточность. Пусть выполнено условие $tin[path_i] \leq up[path_{i+1}]$. Тогда вершина $v = path_i$ является точкой сочленения. Кроме того, это означает, что

  1. либо $tin[path_i] < up[path_{i+1}]$ и тогда ребро $path_i \to path_{i+1}$ — мост, а для любого простого пути из a в b любой мост обязательно содержит оба конца в ответе исходной задачи,
  2. либо $tin[path_i] = up[path_{i+1}]$, тогда ребро $path_i \to path_{i+1}$ находится в другой компоненте рёберной двусвязности, нежели ребро $path_{i-1} \to path_i$, то есть мы пересекли компоненту, и, очевидно, попасть в любую из вершин, достижимых по прямым рёбрам обхода в глубину из вершины $path_{i-1}$ нельзя, ми-

нуя этот переход компонент из-за того, что мы рассматриваем простой путь.

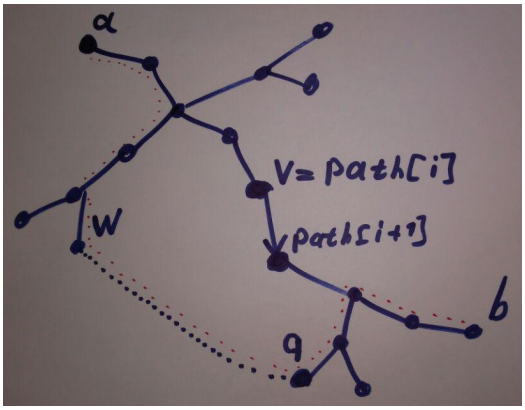

Рис. 2: Необходимость

- Необходимость. Пусть вершина v входит в ответ. Рассмотрим любой простой путь из a в b, построенный на прямых рёбрах обхода в глубину. Этот путь содержит вершину $v = path_i$. Докажем, что $tin[path_i] \leq up[path_{i+1}]$, от противного.

  Пусть $tin[path_i] > up[path_{i+1}]$, тогда существует некая вершина q, из которой есть ребро в вершину w такую, что $tin[w] = up[path_{i+1}]$. Построим новый путь, который не будет содержать вершину v. Обратим внимание, что вершины q и b достижимы из вершины $path_{i+1}$ по прямым рёбрам обхода в глубину, следовательно, можно построить путь от q до b, минуя вершину v. Теперь построим путь от a до q, минуя v. Так как $tin[path_i] > up[path_{i+1}] = tin[w]$, то вершина w встретилась в обходе в глубину раньше v и есть путь от a до w, минуя вершину v. В итоге построили два пути :$a \to w$ и $q \to b$, а вершины w и q соединены ребром. То есть построили путь $a \to b$, минуя вершину v. Противоречие.