1 C. Количество различных путей

Дан невзвешенный неориентированный граф. В графе может быть несколько кратчайших путей между какими-то вершинами. Найдите количество различных кратчайших путей между заданными вершинами. Требуемая сложность O(V+E).

2 Описание алгоритма

Пусть мы идем от вершины b к вершине e. Будем хранить два массива: $depth[V]$ и $wayscount[V]$. В первом будем хранить расстояние от вершины до b или -1, если мы ее еще не посетили, а во втором количество кратчайших путей до нее. Суть алгоритма:

- Начинаем обход в глубину, начиная с b. $wayscount[b] = 1$
- Если мы уже были в вершине, то проверяем, равняется ли ее $depth$ нашему, увеличенному на 1. В таком случае добавляем к ее количеству кратчайших путей наше, кладем ее в очередь. Иначе игнорируем ее и не кладем ее в очередь.
- Если мы в ней не были, то ставим количество путей до нее равный количеству путей из вершины, из которой мы пришли и расстояние на 1 больше. Кладем ее в очередь.
- Ответ равняется $wayscount[e]$

3 Доказательство корректности работы

BFS обрабатывает вершины в порядке их отдаленности от первой вершины. Поэтому, обработав все вершины на расстоянии N все последующие будут на расстоянии не меньше $N + 1$. Таким образом, если мы попадаем из вершины на расстоянии k в вершину на расстоянии $k+1$, то это кратчайший путь, так как он различается не более, чем на единицу. Таким образом, мы посчитаем все кратчайшие пути до всех вершин и $wayscount[e]$ - искомый ответ.

4 Время работы и доп. память

- V - количество вершин, E - количество ребер
- Время работы $O(V + E)$
- Доп. память $O(V + E)$

5 Доказательство времени работы

Время работы следует из времени работы поиска в ширину ($O(V + E)$), который запускается один раз. Доп. память тратится на хранение графа ($O(V + E)$), хранение очереди, дистанций до вершин и количества путей (по $O(V)$).