Билет 44. Определение кратчайшего расстояния в невзвешенном/взвешенном графе.

Def. Если $G = (V, E)$ – невзвешенный граф, то кратчайшее расстояние между u и v:

$$dist(u, v) = \begin{cases} min \text{ число ребер в пути от } u \text{ до } v & \text{если между этими вершинами есть путь} \\ +\infty & \text{если пути между вершинами нет} \end{cases}$$

Def. Взвешенный граф $G = (V, E, w)$, где $w : E \to \mathbb{R}$ – весовая функция.

Def. G – взвешенный граф. Тогда $dist(u, v) = min$ сумма весов на пути от u до v (и снова $+\infty$, если пути нет).

Билет 45. Поиск в ширину: алгоритм bfs с доказательством корректности.

Этот алгоритм находит кратчайшие расстояния от заданной вершины s до всех вершин в невзвешенном графе.

```
vector <vector <int> > g;
vector <int> dist;
dist.assign(n, inf);
dist[s] = 0;
queue <int> q;
q.push(s);

while(!q.empty()) {
  int v = q.front();
  q.pop();
  for(int to : g[v]) {
    if (dist[to] != inf) continue;
    dist[to] = dist[v] + 1;
    q.push(to);
  }
}
```

Асимптотика $O(n + m)$ т.к. каждая вершина побывает в очереди ≤ 1 раз.

Будем доказывать по индукции чуть более сильное утверждение (чем корректность), состоящее из трех частей.

Теорема. *Пусть* $dist[q.front()] = k$

1) Тогда q *имеет вид* $k, \ldots, k, k+1, \ldots, k+1$

2) Все $v : dist(s, v) \leq k$ *либо лежат в очереди, либо уже удалены из нее*

3) Если алгоритм выполнил присваивание $dist[w] = m$*, то* $dist(s, w) = m$

Доказательство. База очевидна. Поэтому докажем сразу переход.

Пусть очередь имеет нужный вид. Достаем первую вершину из очереди – v с $dist[v] = k$. Тогда в конец очереди мы добавим вершины с $dist = k + 1$ просто исходя из работы алгоритма. Значит, очередь будет по-прежнему иметь нужный вид. Мы доказали переход для 1 утв.

Докажем 3. Заметим, что для всех добавленных вершин мы посчитали расстояние верно.

Действительно, если мы рассматриваем ребро $v \to to$, то $dist[to] \le k+1$. Докажем теперь, что меньше, чем $k+1$ быть не может. Пусть это не так. Тогда по второму предположению индукции вершина to уже была обработана, а, значит, не могла быть добавлена в очередь на текущем шаге (потому что уже лежала там или даже была удалена).

Докажем 2. Заметим, что если предыдущая удаленная вершина, как и текущая, имела $dist = k$, то утверждение 2 остается верным и для текущей вершины. Поэтому единственный осмысленный случай – у предыдущей вершины $dist = k$, а у текущей $k+1$.

Докажем, что все вершины с $dist = k+1$ лежат в очереди. Пусть это не так. Тогда есть вершина u с $dist(s,u) = k+1$. Пусть соседняя с ней вершина в этом пути – это вершина p. Тогда $dist(s,p) = k$, и по предположению индукции, p в какой-то момент была удалена из очереди (или она была удалена последней, или до этого). А, значит, u должна была быть добавлена в очередь. Противоречие. Значит, наше предположение неверно. $\square$

Билет 46. Алгоритм $0 - K - bfs$.

Считаем, что наша весовая функция $w : E \to \{0, \dots, k\}$.

Если k не очень большое, то можно модифицировать наш bfs.

Заметим, что $\forall v : dist(s,v) \le k \cdot (n-1) \le kn$, где n – число вершин.

Заводим массив из $kn+1$ очереди. $q[d]$ – очередь вершин, находящихся на расстоянии d от s. А дальше проходимся по нашим очередям в порядке увеличения расстояния и действуем аналогично обычному bfs.

```
vector <int> dist(n, inf)
vector <bool> used(n, false)
dist[s] = 0
q[0].push(s)
for (int d = 0; d <= kn; ++d){
  while(!q[d].empty()){
    int v = q[d].front();
    q[d].pop(); if (used[v]) continue;
    used[v] = true;
    for (edge &e : g[v]) {
      int to = e.to;
      if (dist[to] <= dist[v] + e.w) continue;
      dist[to] = dist[v] + e.w;
      q[dist[to]].push(to);
    }
  }
}
```

Асимптотика $O(kn+m)$.

Билет 47. Двусторонний bfs

Ищем $dist(s,t)$ для заранее заданных s и t.

Идея: с помощью bfs ищем все вершины на расстоянии 1 от s, с помощью bfs ищем вершины на расстоянии 1 от t (на графе с "обратными"ребрами), ищем вершины на расстоянии 2 от s и вершины на расстоянии 2 от t. Будем так продолжать, пока не возникнет какая-то вершина, которая встретилась и для s, и для t. По сути, мы как будто параллельно запускаем bfs в s и t.

Пусть mid – первая точка, которая будет найдена с обеих сторон. Докажем, что через нее проходит кратчайший путь. Пусть это не так. Тогда есть какой-то путь между s и t длины меньше, чем $dist(s, mid) + dist(mid, t)$. Заметим также, что $dist(s, mid)$ и $dist(mid, t)$ отличаются не более, чем на 1 в силу нашего алгоритма. Возьмем вершину v на расстоянии $dist(s, mid) - 1$ от s. Тогда $dist(v, t) \leq dist(mid, t)$. Но тогда mid – не первая вершина, которая встретилась и у s, и у t. Противоречие. Значит, наше предположение неверно.

В общем случае асимптотика как у обычного bfs, но в частных случаях он выгоднее. Например, когда количество вершин на каждом уровне растет экспоненциально (например, из каждой вершины исходит по 3 ребра, или что-то в этом духе).