



N1



The screenshot shows a terminal window with a dark theme. At the top, there are three colored window control buttons (red, yellow, green). To the right of them is the text "C++ DijkstraSSSP". The main area contains the following C++ code:

```
void Graph::dijkstraSSSP(size_t start) {
    std::vector<size_t> dist;
    std::vector<size_t> path;

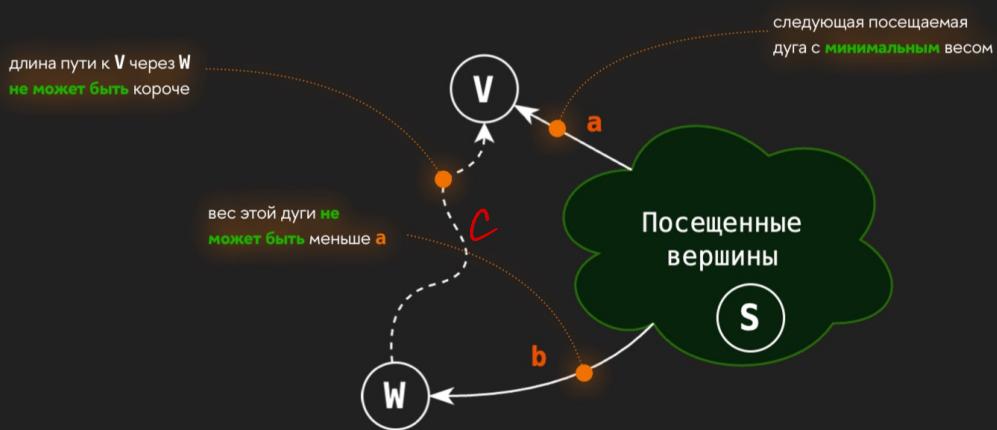
    инициализация расстояний в вершинах;

    while (есть непосещенные вершины) {
        найти вершину v с наименьшим значением dist;

        for (u : adjacent[v] И u не посещена) {
            if (dist[u] > weight(v, u) < dist[v]) {
                dist[u] = dist[v] + weight(v, u)
                path[u] = v
            }
        }
    }
}
```

Единственное отличие - замена  
словесных на условие в if  
Ограничение на граф:

# Жадный алгоритм Дейкстры работает?



В обычной алгоритме Дейкстры  
у нас такой инвариант:  $a \leq b + c$

В модифицированном сложение заменяется  
на умножение, так что будем  $a \leq bc$   
Это достигается лишь при  $a, b, c \geq 1$ .

Потому ограничение на граф такое  
все каскадного ребра  $\geq L$ . Отрезанные  
вершины не попадают в граф могут

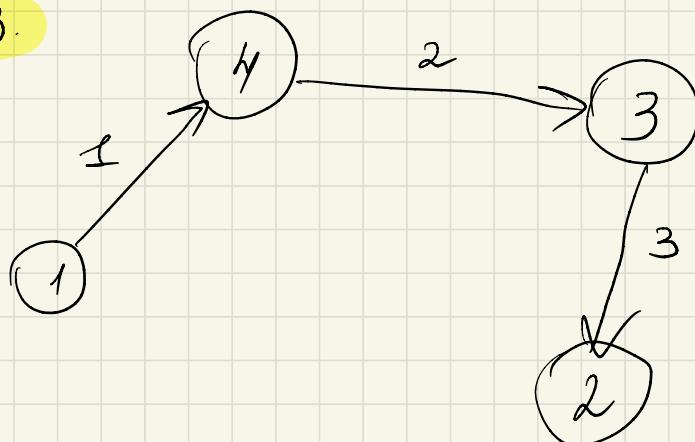
Учиться идти путь до вершины V  
через ВС, что противоречит извращену  
согласованности Декартовы для таких градов  
принесенных аргументов Рорда-Беллмана

N2.

Заметим, что для каждого матрица  
крайнейших расстояний существует  
более одного града за некоторыми  
случаи, так что никакой единой  
загадки смысла не определено, но  
можно можно сказать, что решения  
будут являться градами, матрицы  
которого являются загадкой  
матрица крайнейших расстояний  
Восстановленный град будет находиться  
или в случае, когда это матрица

Задача некорректна ибо когда  
путь между какими либо двумя  
вершинами бесконечно мал ( $-\infty$ ).  
Во втором случае проверяется, как  
и где различаются текущий мин. веса.

N3.



	1	2	3	4
1	0	100	100	1
2	100	0	100	100
3	100	3	0	100
4	100	100	2	0

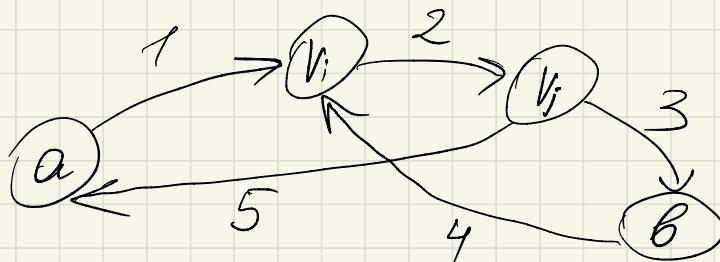
i	j	k	dist[i][j]
1	1	1	0
1	1	2	0
1	1	3	0
1	1	4	0
1	2	1	$+\infty$
1	2	2	$+\infty$
1	2	3	$+\infty$
1	2	4	$+\infty$

$dist[i][j] = \min ($   
 $dist[i][j],$   
 $dist[i][k] + dist[k][j])$

Длинная трассировка алгоритма  
занимает еще 56 строк, так что  
приводить её не буду. Так  
быстро, что алгоритм считает, что  
не существует пути между вершина-  
ми ① и ②, хотя он есть и его  
длина 6 (минимальная длина).

Длиннейшая трассировка более не  
будет называться длиннейшей  $dist[1][2]$ ,  
так как будут появляться другие  
пары  $i, j$ .

N4. Да, такой граф определений можно:



$a \rightarrow v_i \rightarrow v_j \rightarrow b$  - кратчайший путь из  
 $a$  в  $b$ , содержит ребро  $(v_i, v_j)$  (бес 6)

$b \rightarrow v_i \rightarrow v_j \rightarrow a$  - кратчайший путь из  
 $b$  в  $a$ , содержит ребро  $(v_i, v_j)$  (бес 11)

### Характеристика данного графа:

вершина  $a$  и вершины нашего особого ребра  $(v_i, v_j)$  могут содержаться в каком-либо цикле (при этом в цикле может быть более 3 вершин)

Напомним, что  $v_i, v_j$  также содержатся в каком-либо цикле

Для доказательства следует отметить эквивалентность  $P$ :  $(x, y) \in P \Leftrightarrow$   
 $\Leftrightarrow$  существует путь из  $x$  в  $y$

# Свойства отношения $P$ ,

по умолчанию

1. Аntирефлексивность;  $\forall (x, x) \notin P$ , т.к.  
т.к. узлы не могут иметь себя членом
2. Из  $(x, y) \in P \Rightarrow (y, x) \notin P$ , т.к.  
не гарантируется наличие обратного  
ребра  $\Rightarrow P$  не является симметричным  
 $\rightarrow$  не-симметричность
3.  $\forall x, y, z \in V : ((x, y) \in P \wedge (y, z) \in P) \Rightarrow$   
 $\Rightarrow (x, z) \in P$  — транзитивность, очевидно,  
выполняется

## Доказательство:

Если узел  $(V_i, V_j)$  лежит на  
краю графа и имеет форму  $a b b \Rightarrow$   
 $\Rightarrow (a, V_i), (a, V_j), (V_j, b), (V_i, b) \in P$

Также  $(V_i, V_j)$  лежит на краю графа и имеет форму  $b a a \Rightarrow$

$\Rightarrow (b, v_i), (b, v_j), (v_i, a), (v_j, a) \in P$

$(v_i, v_j) \in P$ , m.k.  $(v_i, v_j)$  - ребро (ориентированное)

Таким образом,  $(b, v_i), (v_i, v_j), (v_j, b) \in P \Rightarrow$

$\Rightarrow$  no транзитивности  $(b, b) \in P \Rightarrow$

$\Rightarrow$  m.k. нечестив, но мы нашли цикл, который содержит ребро  $(v_i, v_j)$

Аналогично:  $(a, v_i), (v_i, v_j), (v_j, a) \in P \Rightarrow$

$\Rightarrow (a, a) \in P \Rightarrow a$  содержится в цикле с ребром  $(v_i, v_j)$ . так

Ограничения применения известных алгоритмов:

- ограничение на деякетру, если есть хотя бы одно отрицательное ребро, в таком случае между использованием алгоритм Bellman-Ford или Dijkstra

• В задаче есть максимальный один член, так что если для шир. веса, то необходимо использовать еще алгоритм Беллмана-Форда, чтобы алгоритм Флойда-Уоршелла с модифицированной на бинарной форме отрицательного веса.