Алгоритмы на графах

Поиск по графу

Задан граф $G = (V, E)$, $A(v), v \in V$ — списки смежности

Найти компоненты связности.

Алгоритм ПОИСК(v)

1. $Q := \{v\}$, пометить v.
2. До тех пор пока $Q \neq \varnothing$ выполнять
   - 2.1. Пусть $x \in Q$; удалить x из Q;
   - 2.2. Для всех $y \in A(x)$:
     если у не помечена, то добавить y в Q и пометить y

Утверждение. Алгоритм ПОИСК помечает все вершины графа, достижимые из v, за время $O(|E|)$.

Доказательство. Пусть V_1 — множество вершин, достижимых из v

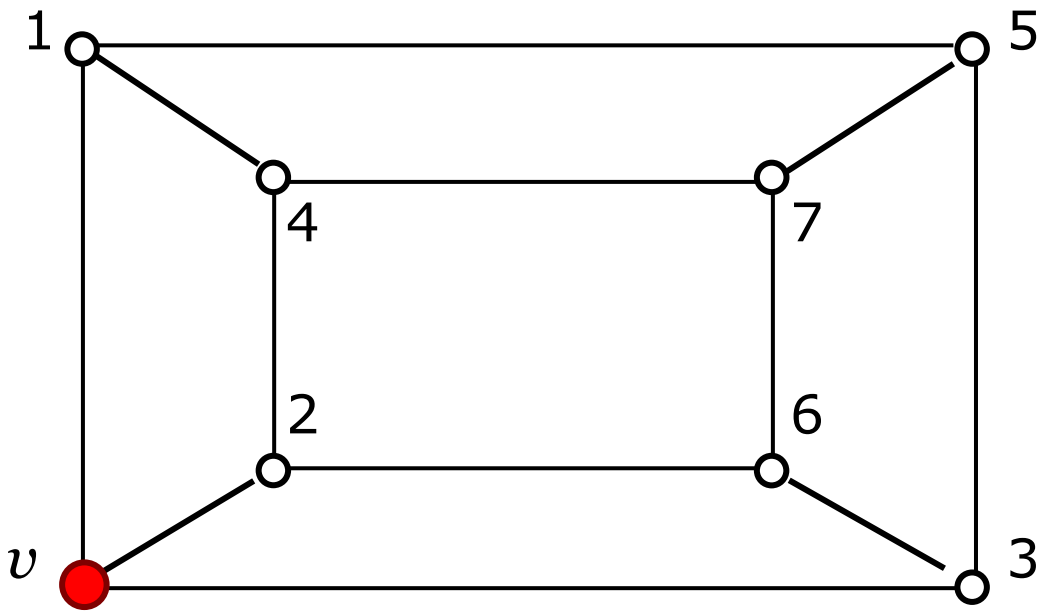
1. Для записи v требуется время $O(1)$.
2. Работа с множеством Q. Добавление и удаление элементов производится $2|V_1|$ раз. Если Q — очередь или стек, то каждое включение или исключение требует $O(1)$ времени.
3. Поиск по спискам смежности. Каждый элемент в списке просматривается не более одного раза. Всего $2|E|$ элементов.

Суммарная оценка — $O(|E|)$.

Поиск в ширину

Множество Q организовано в виде очереди:

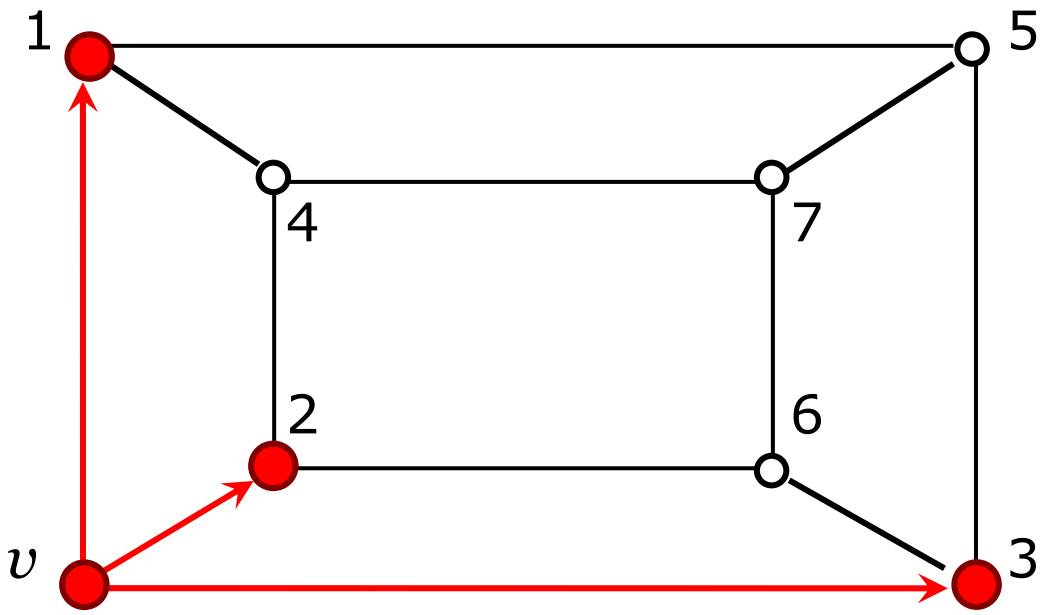
первым пришел — первым ушел.



Поиск в ширину

Множество Q организовано в виде очереди:

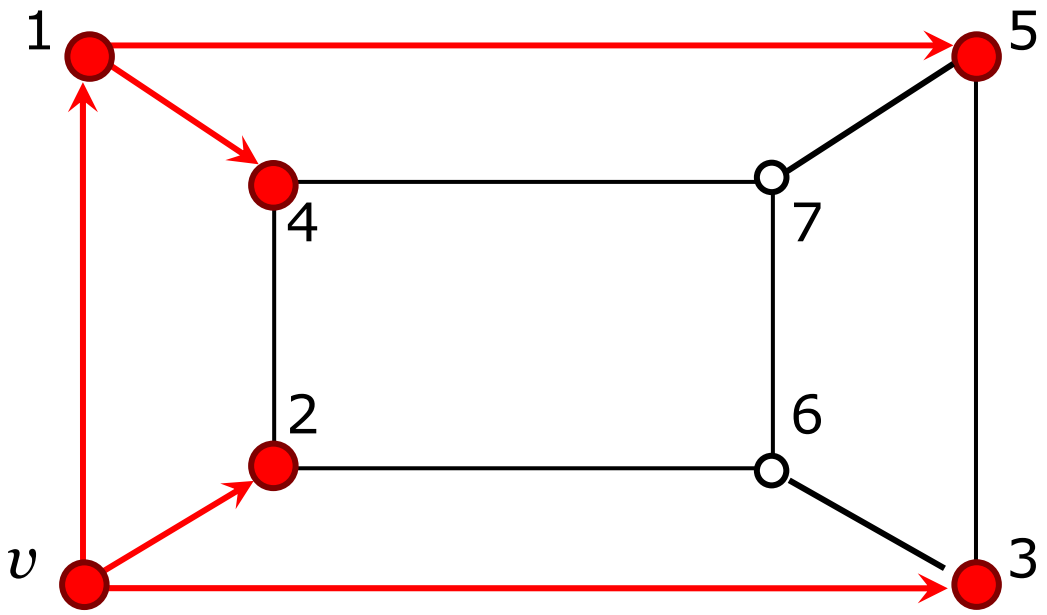
первым пришел — первым ушел.



Поиск в ширину

Множество Q организовано в виде очереди:

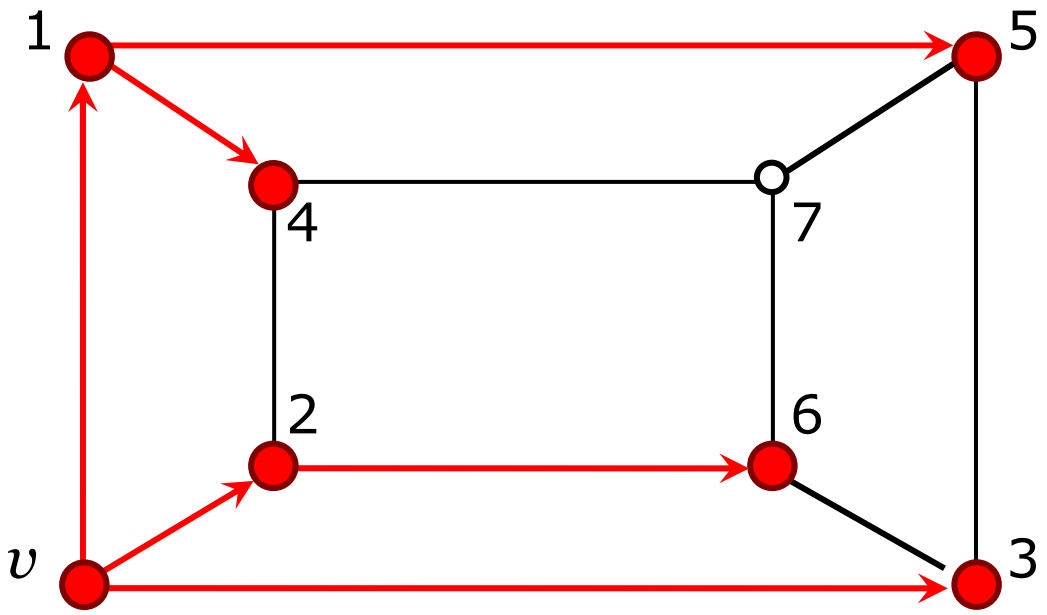
первым пришел — первым ушел.



Поиск в ширину

Множество Q организовано в виде очереди:

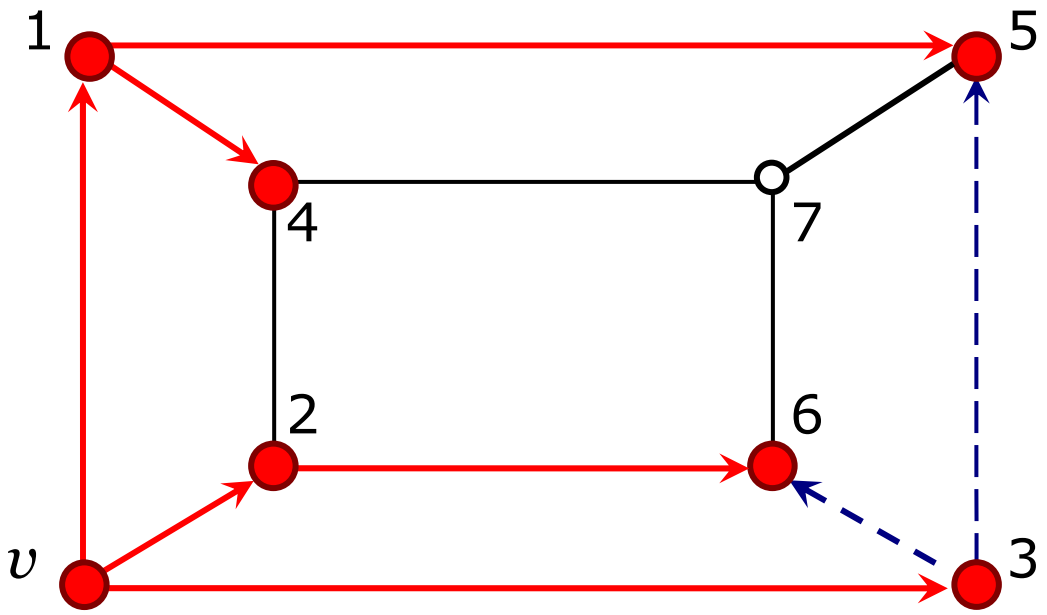
первым пришел — первым ушел.



Поиск в ширину

Множество Q организовано в виде очереди:

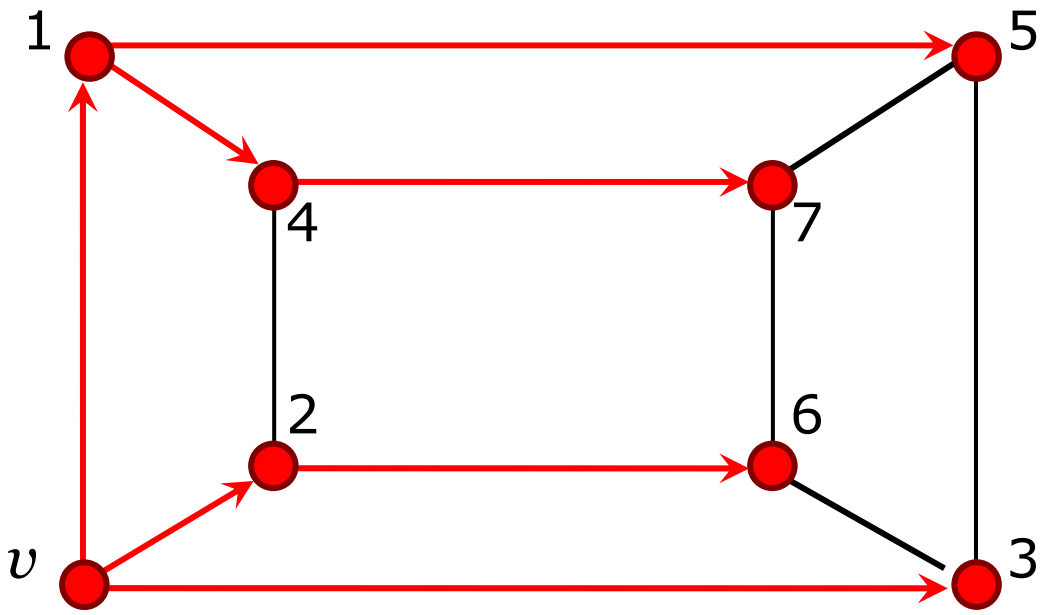
первым пришел — первым ушел.



Поиск в ширину

Множество Q организовано в виде очереди:

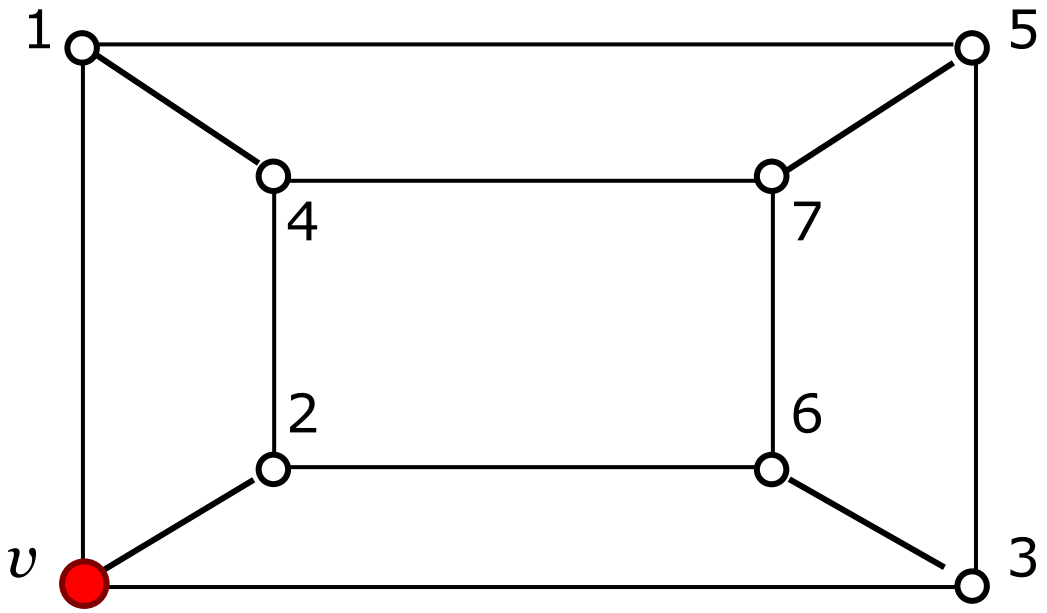
первым пришел — первым ушел.



Поиск в глубину

Множество Q организовано в виде стека:

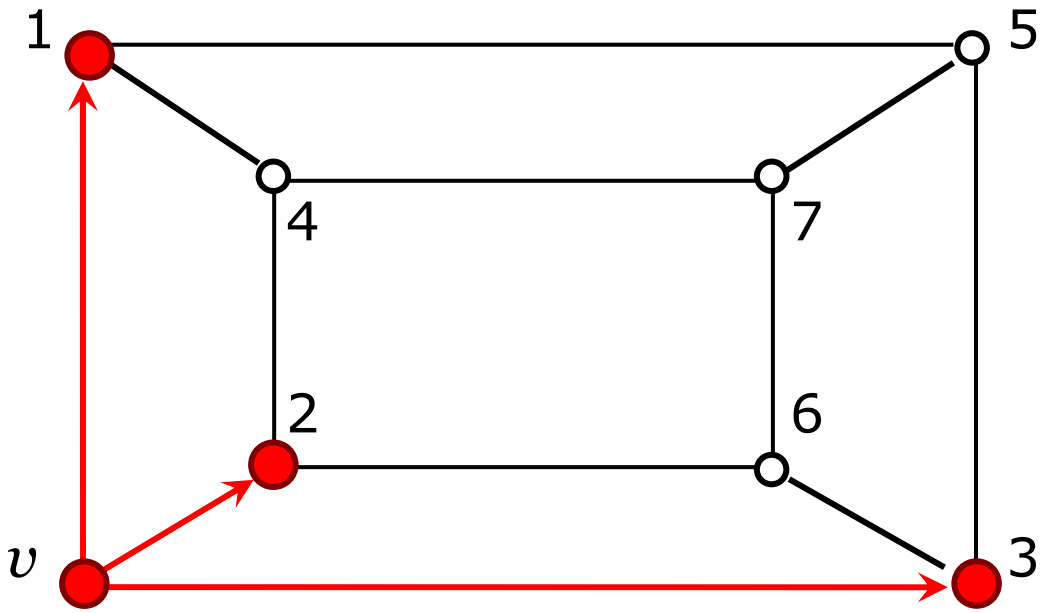
последним пришел — первым ушел.



Поиск в глубину

Множество Q организовано в виде стека:

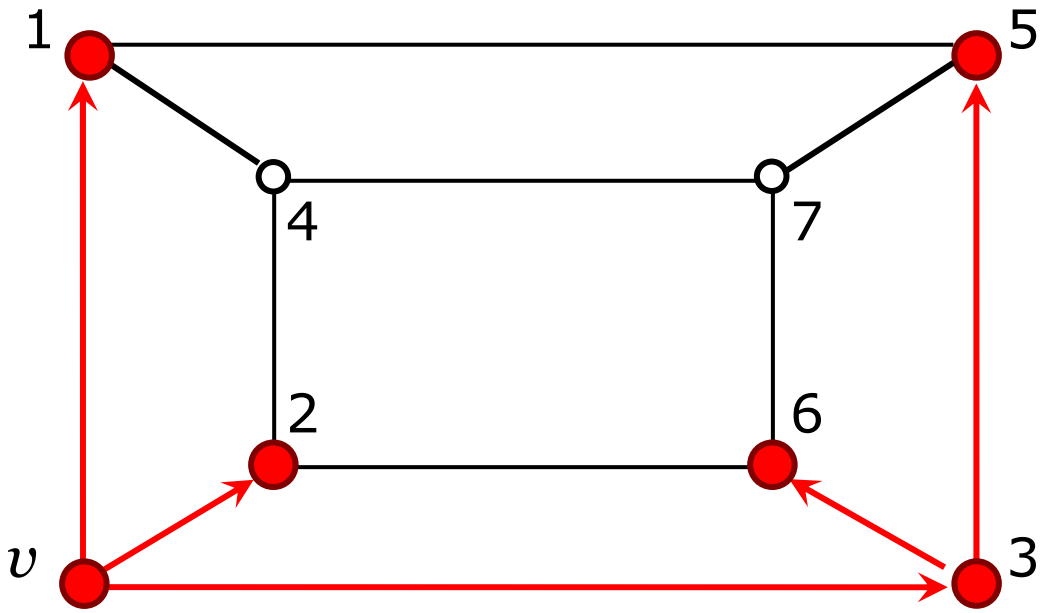
последним пришел — первым ушел.



Поиск в глубину

Множество Q организовано в виде стека:

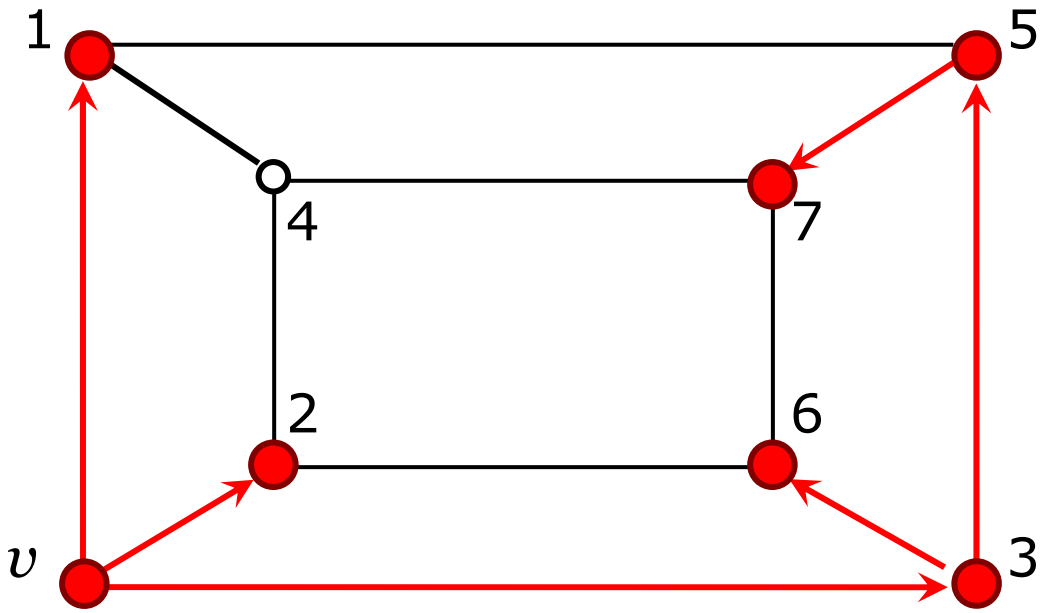
последним пришел — первым ушел.



Поиск в глубину

Множество Q организовано в виде стека:

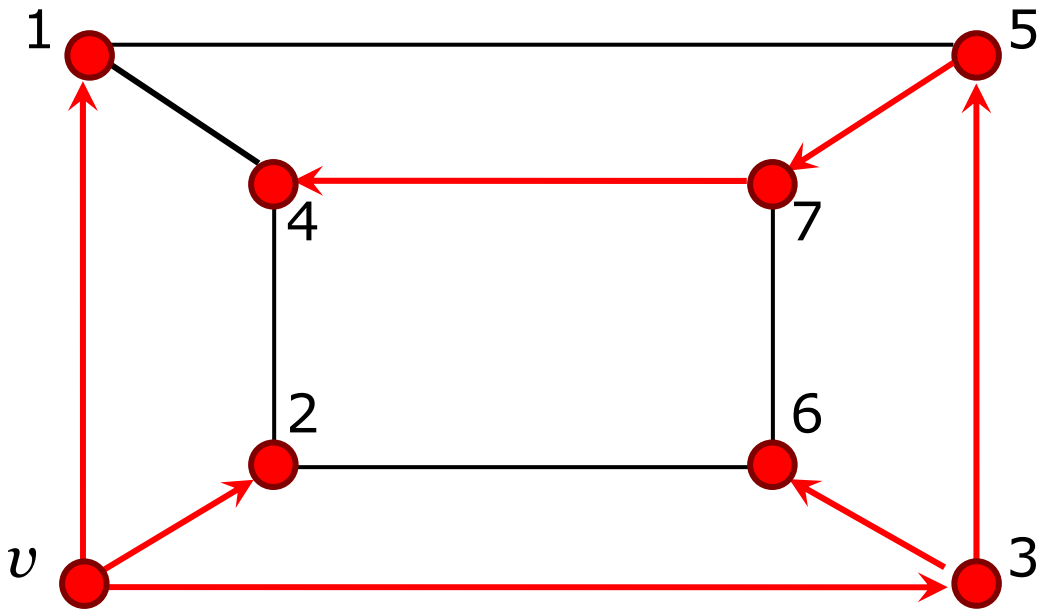
последним пришел — первым ушел.



Поиск в глубину

Множество Q организовано в виде стека:

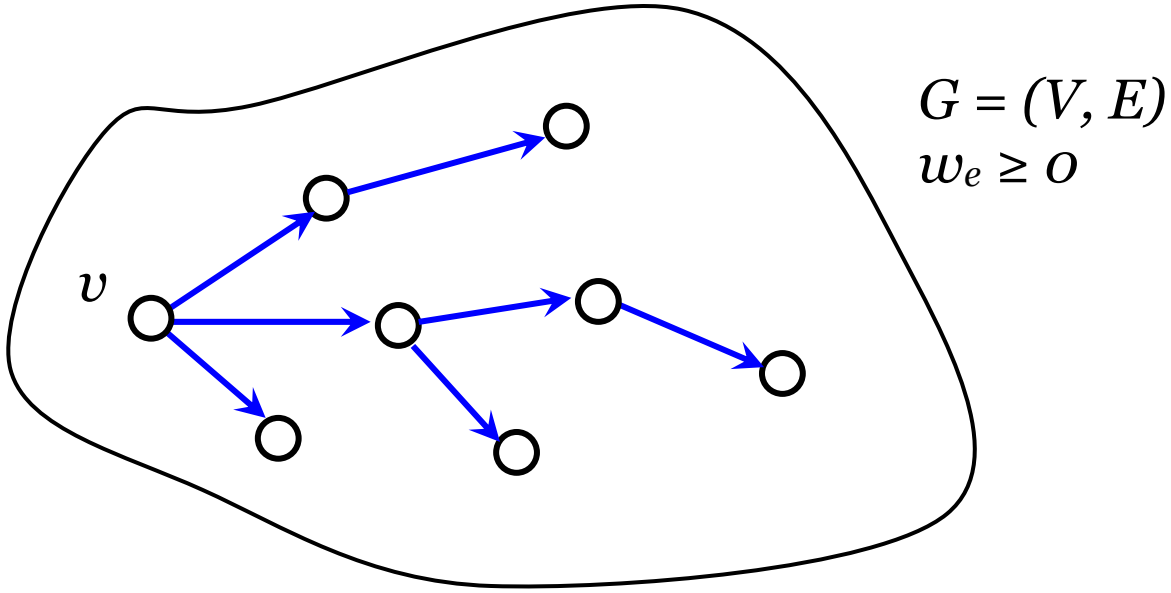
последним пришел — первым ушел.



Задача о кратчайшем пути

Задан ориентированный граф $G=(V, E)$, $w_e \geq 0, e \in E$ — вес дуги.

Найти кратчайшие расстояния от заданной вершины v до остальных вершин.



Алгоритм Дейкстры

1. Положить $W=\{v\}$; $\rho(x) = 0$; $p(v) = \varnothing$.
2. Для всех $y \in V \setminus \{v\}$:
   2.1 $\rho(y) := w_{vy}$;
   2.2 $p(y) := v$;
3. До тех пор пока $W \neq V$ выполнять
   3.1 Найти такую вершину $x \in V \setminus W$, что $\rho(x) = \min\{\rho(y) \mid y \in V \setminus W\}$;
   3.2 Положить $W := W \cup \{x\}$;
   3.3 Для всех $y \in V \setminus W$
   { $z := \rho(y)$;
   $\rho(y) := \min\{\rho(y), \rho(x) + w_{xy}\}$;
   если $\rho(y) < z$, то $p(y) := x$ }.

Утверждение. Алгоритм Дейкстры находит кратчайшие пути из вершины v до каждой из остальных вершин за время $O(|V|^2)$.

Доказательство. Покажем, что на каждой итерации:

а) $\forall x \in V$ величина $\rho(x)$ равна длине кратчайшего из путей от v до x, все промежуточные вершины которых принадлежат W.

б) $\forall y \in W$ величина $\rho(y)$ равна длине кратчайшего из путей от v до y.

Так как в конце работы алгоритма $W = V$, то из б) следует, что $\rho(x)$ — вектор кратчайших расстояний.

Проводим индукцию по числу шагов алгоритма

1. При $W = \{v\}$ утверждения а), б) верны.
2. Пусть а), б) верно для некоторого шага.

На шаге 3.1. мы выбираем $x \in V \setminus W$ такую, что $\rho(x) = \min\{\rho(y) | \, y \in V \setminus W\}$.

Предположим, что $\exists$ путь $(v, v_1, \ldots, v_t, x)$, длина которого меньше $\rho(x)$. Тогда из а) следует, что в этом пути есть вершина $v_i \notin W$. Если таких несколько, выберем вершину с наименьшим номером. Тогда $\rho(v_i) \le$ длина $(v, v_1, \ldots, v_i) \le$ длина $(v, v_1, \ldots, v_t) \le \rho(x)$, что противоречит выбору x. Значит $\rho(x)$ — длина кратчайшего пути от v до x и б) будет выполняться после добавления x к W.

На шаге 3.3 после пересчета $\rho(y) = \min\{\rho(y), \rho(x) + w_{xy}\}$ получим а), так как любой путь в y идет либо через x, либо мимо x.

Итак а), б) верно. Оценим трудоемкость. Цикл 3 требует $O(|V|)$ итераций. На каждой итерации 3.1 — $O(|V|)$; 3.2 — $O(1)$; 3.3 — $O(|V|)$.

Итого: $O(|V|^2)$. ■

Алгоритм Флойда – Уоршелла

Задан ориентированный граф $G=(V, E)$, $w_e \geq 0$, $e \in E$.

Найти кратчайшее расстояние для каждой пары вершин.

Определение. Для квадратной матрицы (d_{ij}) *операцией треугольника* относительно j называется пересчет $d_{ik} = \min\{d_{ik}, d_{ij} + d_{jk}\}$ по всем $i,k \neq j$.

Утверждение. Пусть c_{ij} — длина дуг орграфа $G=(V, E)$ и

$$d_{ij} = \begin{cases} c_{ij}, & \text{если } i \neq j \\ 0, & \text{если } i = j \end{cases}.$$

Если выполнить над матрицей (d_{ij}) операцию треугольника последовательно для $j = 1, 2, \ldots, |V|$, то в полученной матрице каждый элемент d_{ik} равен длине кратчайшего пути из i в k.

Доказательство. Покажем, что для каждого j после выполнения операций треугольника $t = 1, 2, \ldots, j$ элемент d_{ik} $\forall i, k$ равен длине кратчайшего пути из i в k среди всех путей, промежуточные вершины которых имеют номера не больше j.

Для $j = 1$ утверждение очевидно.

Пусть оно верно для $j = t - 1$, и проводится операция для t: $d_{ik} = \min\{d_{ik}, d_{it} + d_{tk}\}$. Рассмотрим подграф G' орграфа G на вершинах $\{1, 2, \ldots, t, i, k\}$. Если кратчайший путь из i в k в G' не проходит через t, то минимум достигается на первом аргументе и утверждение верно. Если же он проходит через t, то $d_{it} + d_{tk} \le d_{ik}$, а по предыдущему предположению d_{it} и d_{tk} — длины кратчайших путей из i в t и из t в k по вершинам с номерами не более t. ■

Алгоритм

1. Для всех $i\,j$: $d_{ij} := c_{ij}$;
2. Для всех i : $d_{ii} := 0$;
3. Для всех $i\,j$: $e_{ij} := 0$;
4. Для всех j :

   для всех $i \neq j$ и для всех $k \neq i,\ k \neq j$:

   4.1. $z := d_{ik}$

   4.2. $d_{ik} := \min \{ d_{ik};\ d_{ij} + d_{jk} \}$

   4.3. если $d_{ik} < z$, то $e_{ik} := j$.

Время $O(|V|^3)$.

Алгоритм работает корректно, даже если есть дуги отрицательной длины, но нет контуров отрицательной длины.

Распределительная задача

Задано:

n — число предприятий;

Y — количество единиц некоторого ресурса;

$f_k(x)$ — количество продукции, которое будет произведено на k-м предприятии, если в него будет вложено x единиц ресурса (монотонно неубывающая функция).

Требуется: максимизировать объем продукции

$$f_1(x_1) + \ldots + f_n(x_n) \to \max \quad (1)$$

$$x_1 + \ldots + x_n \le Y \quad (2)$$

$$x_i \ge 0, \text{ целые}, i = 1,\ldots n. \quad (3)$$

Идея динамического программирования (ДП)

Метод ДП (Р. Беллман, В.С. Михалевич, Н.З. Шор) можно трактовать как алгоритмическую версию рассуждений по индукции.

Пусть $s_k(y)$, $1 \le k \le n$, $0 \le y \le Y$, — оптимальное значение целевой функции задачи (1) – (3), где n заменено на k, Y заменено на y.

Требуется найти $s_n(Y)$ и набор переменных, на котором достигается это значение.

Теорема 1. Пусть $f_1, \ldots, f_n$ — монотонно неубывающие функции.

Тогда справедливы следующие *рекуррентные соотношения:*

$$s_1(y) = f_1(y), \ 0 \le y \le Y; \tag{4}$$

$$s_k(y) = \max\{s_{k-1}(y-x) + f_k(x) \mid 0 \le x \le y\}, \ 2 \le k \le n, \ 0 \le y \le Y, \tag{5}$$

Доказательство: Соотношение (4) очевидно. По определению

$$s_k(y) \ge \max\{s_{k-1}(y-x) + f_k(x) \mid 0 \le x \le y\}.$$

Пусть теперь $(x_1^*,\ldots,x_k^*)$ — такой вектор, что $x_1^* + \ldots + x_k^* \le y$ и

$$s_k(y) = f_1(x_1^*) + \ldots + f_k(x_k^*).$$

Поскольку $s_{k-1}(y - x_k^*) \ge f_1(x_1^*) + \ldots + f_{k-1}(x_{k-1}^*)$, имеем

$$s_k(y) = f_1(x_1^*) + \ldots + f_k(x_k^*) \le s_{k-1}(y - x_k^*) + f_k(x_k^*). \blacksquare$$

Алгоритм ДП вычисляет множество $S_k = \{s_k(y) \mid 0 \le y \le Y\}$, $k = 1,\ldots, n$, с помощью соотношений (4) и (5), где на каждом шаге оптимизируется ровно одна переменная.

Процесс вычисления $S_1, \ldots, S_n$ называется *прямым ходом* алгоритма.

Число операций $\approx Y^2 n$

Память $\approx Y n$.

y	$S_1(y)$	$S_2(y)$	$\cdots$	$S_n(y)$
0				
1				
2				
$\vdots$				
Y				$S_n(Y)$

При *обратном ходе* алгоритма вычисляются значения $(x_n^*,\ldots, x_1^*)$, с учетом того, что уже известны $S_k(y)$. Например, x_n^* определяется из уравнения $s_n(Y) = f_n(x_n^*) + s_{n-1}(Y - x_n^*)$ и так далее.

Число операций $\approx Yn$. Память $\approx Y n$.

Характеристики алгоритмов

Для оценки качества алгоритмов будем использовать два параметра:

T_A — *трудоемкость* (число элементарных операций алгоритма A);

Π_A — требуемый *объем памяти*.

Элементарная операция — одна из арифметических операций: сложение, вычитание, умножение, деление или логическая операция сравнение двух чисел.

Нас будет интересовать зависимость параметров алгоритма от длины записи исходных данных задачи с точностью до порядка величин.

Пример: При $T = 3/2\, n^2$, будем писать $T = O(n^2)$ или $T \approx n^2$.

Полиномиальные алгоритмы

Определение. Алгоритм A называют *полиномиальным*, если его трудоемкость T_A ограничена полиномом от длины записи исходных данных, то есть существует константа $c > 0$ и натуральное число k такие, что $T_A \le c\,L^k$, где L — длина записи исходных данных.

Пример: Пусть $f_i(x_i) = a_i\,x_i$, тогда $L = \sum_{i=1}^{n} \log a_i + \log Y$,

но $T_{\text{ДП}} = O(Y^2 n)$, то есть алгоритм ДП не является полиномиальным.

Обобщим задачу (1)–(3):

$$f_1(x_1) + \ldots + f_n(x_n) \to \max \qquad (1')$$

$$h_1(x_1) + \ldots + h_n(x_n) \le Y \qquad (2')$$

$$a_i \ge x_i \ge 0, \text{ целые}, \quad i = 1,\ldots n. \qquad (3')$$

Если $h_i(x)$ — целочисленные монотонно неубывающие функции, то вместо (4)–(5) можно использовать следующие *рекуррентные соотношения:*

$$s_1(y) = f_1(x^*), \text{ где } x^* = \max\{x \le a_1 \mid h_1(x) \le y\}, \quad 0 \le y \le Y; \qquad (4')$$

$$s(y) = \max_{\{x \le a_k \mid h_k(x) \le y\}} \{f_k(x) + s_{k-1}(y - h_k(x))\}, \quad 2 \le k \le n, \quad 0 \le y \le Y. \qquad (5')$$

Упражнение 1. Доказать справедливость соотношений (4′)–(5′).

Обратная задача — поиск наименьших затрат на получение заданного количества продукции:

$$h_1(x_1) + \ldots + h_n(x_n) \to \min \quad (6)$$

$$f_1(x_1) + \ldots + f_n(x_n) \geq D \quad (7)$$

$$a_i \geq x_i \geq 0, \text{ целые}, \ i = 1,\ldots n. \quad (8)$$

Если $f_k(x)$ — целочисленные монотонно неубывающие функции, то для решения задачи (6)–(8) можно использовать идеи динамического программирования.

Пусть $f_i^{-1}(d) = \min\{0 \le x \le a_i \mid f_i(x) \ge d\}$.

Для $1 \le k \le n$, $0 \le d \le D$ обозначим через $t_k(d)$ — оптимальное решение задачи (6)–(8), в которой n заменено на k, а D заменено на d.

Требуется найти $t_n(D)$.

Рекуррентные соотношения

$$t_1(d) = \begin{cases} \infty, & \text{если } f_1(a_1) < d, \\ h_1(f_1^{-1}(d)), & \text{если } f_1(a_1) \ge d, \end{cases} \quad 0 \le d \le D, \tag{9}$$

$$t_k(d) = \min\{t_{k-1}(d - f_k(x)) + h_k(x) \mid 0 \le x \le a_k, x \le f_k^{-1}(d)\}, \tag{10}$$

$$k \ge 2, \ 0 \le d \le D.$$

Упражнение 2. Доказать справедливость соотношений (9)–(10).

Теорема 2: Предположим, что D — наибольшее число, для которого оптимальное значение целевой функции задачи (6)–(8) не превосходит Y. Тогда оптимальное значение целевой функции задачи (1′)–(3′) равно D.

Доказательство: Пусть D удовлетворяет условию теоремы и $(x_1^*,\ldots,x_n^*)$ — соответствующее решение задачи (6)–(8).

Значит

$$f_1(x_1^*)+\ldots+f_n(x_n^*)\geq D \quad \text{и} \quad h_1(x_1^*)+\ldots+h_n(x_n^*)\leq Y.$$

Следовательно, D не превосходит оптимального решения D_1 задачи (1′)–(3′). Если бы D_1 было больше D, то решение задачи (6)–(8), в которой D заменено на D_1, тоже не превышало бы Y, что противоречит максимальности D. ■