Лекция 12. Представление графов. Обход графов: BFS && DFS

Казахский национальный университет имени аль-Фараби

Кафедра компьютерных наук

Алгоритмы и структуры данных

Лектор: старший преподаватель, PhD, Кенжебек Ержан Ғалымжанұлы

Цель и план лекции

Цель лекции

Изучить способы представления графов (список смежности, матрица смежности, список рёбер) и алгоритмы обхода графа в глубину (DFS) и в ширину (BFS).

План лекции

1. Ориентированные и неориентированные графы, терминология
2. Список смежности
3. Матрица смежности, сравнение представлений
4. Список рёбер
5. Поиск в глубину (DFS)
6. Поиск в ширину (BFS), реализация на C++

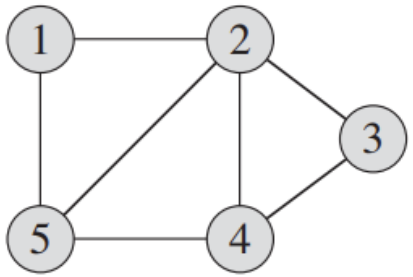
Представление графов

- Есть два стандартных способа представления графа *G = (V, E):*
  - Набор списка смежных вершин (adjacency-list)
  - Матрица смежности (adjacency matrix)

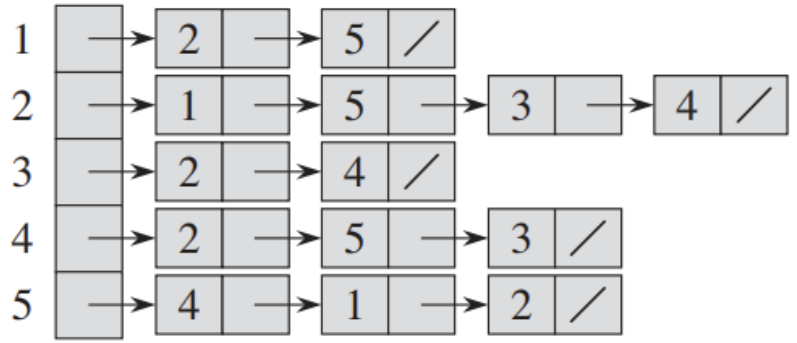
- Применяются как для ориентированных, так и для неориентированных

- Когда и какое представление графов следует использовать?
  - Предпочтительным является представление с помощью списков смежности, поскольку оно обеспечивает представление разреженных (sparse) графов $|E| < |V^2|$
  - Представление с помощью матрицы смежности предпочтительнее когда мы имеем плотные (dense) графы, когда значение $|E|$ близко к $|V^2|$ или при необходимости быстро определить существует ли ребро между вершинами

Неориентированный граф



(a)

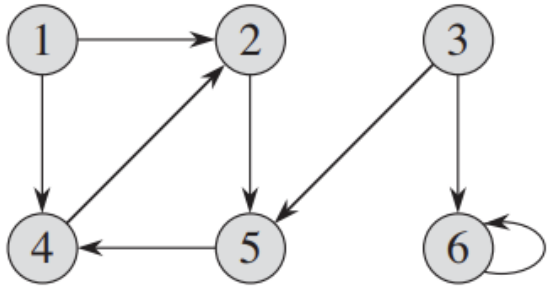


(b)

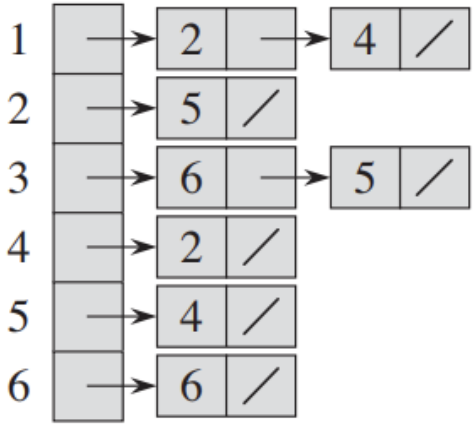
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

(c)

Ориентированный граф



(a)



(b)

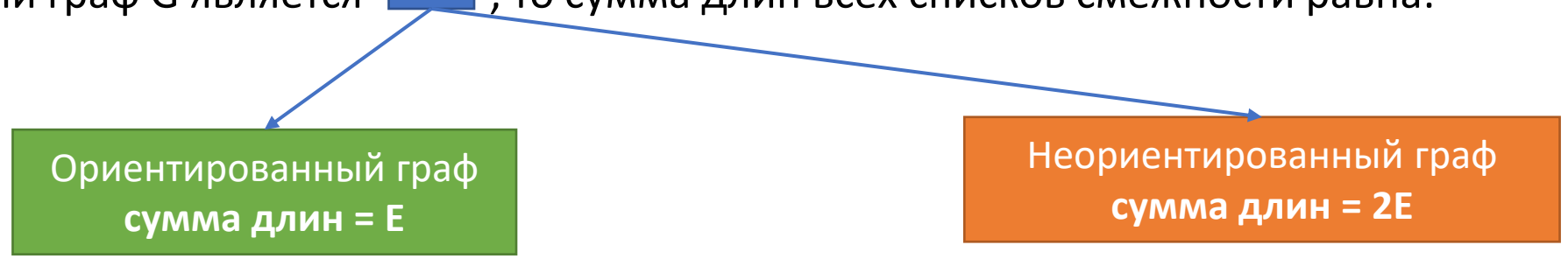
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

(c)

Список смежности

- Представление графа G = (V, E) в виде списка смежности состоит из массива Adj из |V| списков, по одному для каждой вершины из **V** . Для каждой вершины $\boldsymbol{u} \in \boldsymbol{V}$ список смежности ***Adj[u]*** содержит все вершины $\boldsymbol{v}$, для которых существует ребро $(u, v) \in \boldsymbol{E}$. То есть ***Adj[u]*** состоит из всех вершин, смежных с $\boldsymbol{u}$ в графе G. (В качестве альтернативы он может содержать указатели на эти вершины.)

- Если граф G является , то сумма длин всех списков смежности равна:



- Объем памяти для обеих графов в виде списке смежности требует: **O(V+E)**

Недостаток представления в виде списка смежности

- Потенциальный недостаток представления в виде списка смежности заключается в том, что он не обеспечивает более быстрого способа определить, является ли данное ребро ***(u,v)*** в графе, чем поиск в списке смежности ***Adj[u]***.
- **Представление графа матрицей смежности устраняет это преимущество, но ценой использования асимптотически большего объема памяти.**

Матрица смежности

- Представление графа G = (V,E) с помощью **матрицы смежности (adjacency matrix representation)** предпологает что вершины пронумерированы в некотором произвольном порядке числами 1,2,…,|V|. Матрица A = a_{ij} размером |V| x |V|:

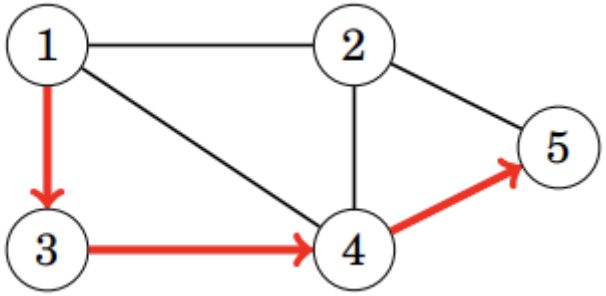
- $a_{ij} = \begin{cases} 1, & if \quad i,j \in E \\ 0, & else \end{cases}$

- Матрица смежности графа требует памяти O(V^2) в независимости от количества ребер графа

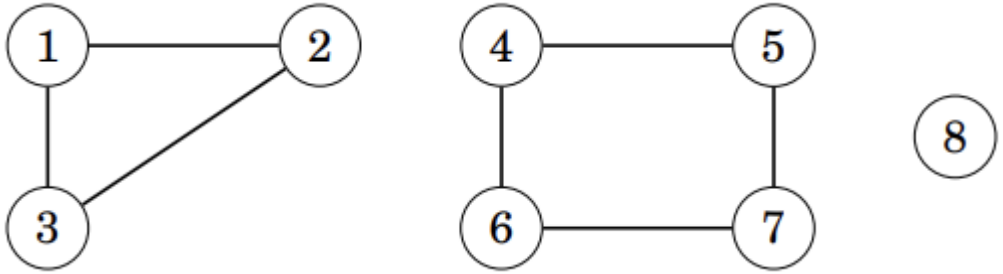
- Оба представление графов можно использовать для взвешенных графов

Terminology

- Cycle:



- Connectivity, example components 3:

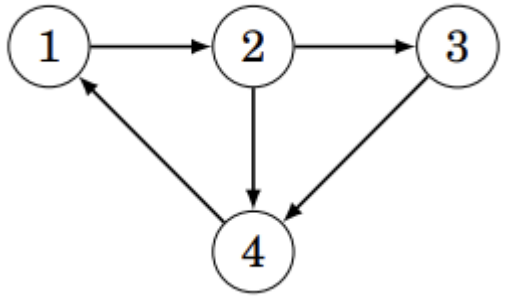


- Edge directions and weights

- V – vertex, E – edges

Список смежности (C++)

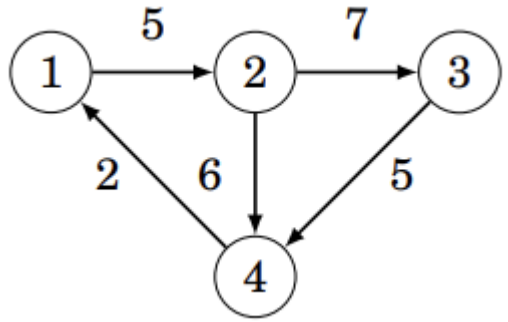
```
vector<int> adj[N];
adj[1].push_back(2);
adj[2].push_back(3);
adj[2].push_back(4);
adj[3].push_back(4);
adj[4].push_back(1);
```



Список смежности с взвешенным графом (C++)

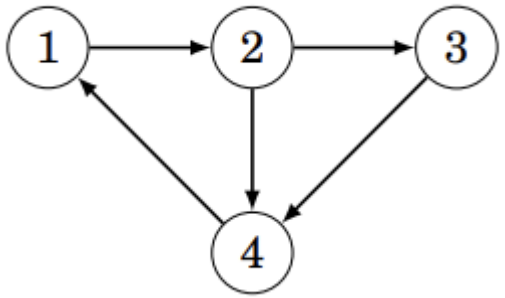
```
vector<pair<int,int>> adj[N];
adj[1].push_back({2,5});
adj[2].push_back({3,7});
adj[2].push_back({4,6});
 adj[3].push_back({4,5});
adj[4].push_back({1,2});

for (auto u : adj[s]) {
   // process node u
}
```

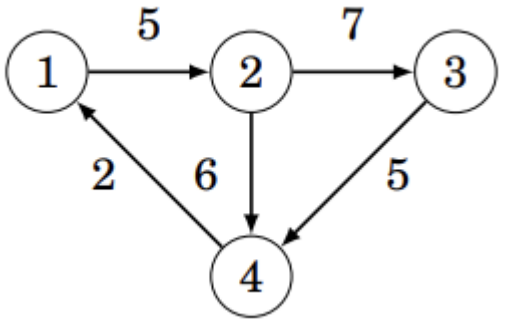


Матрица смежности (C++)

- int adj[N][N];

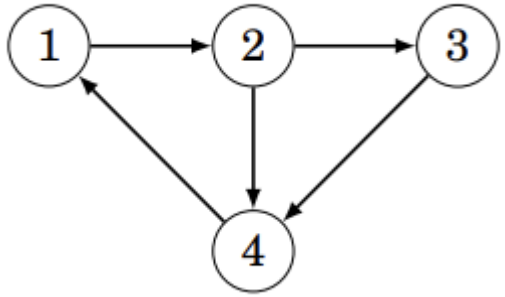


	1	2	3	4
1	0	1	0	0
2	0	0	1	1
3	0	0	0	1
4	1	0	0	0

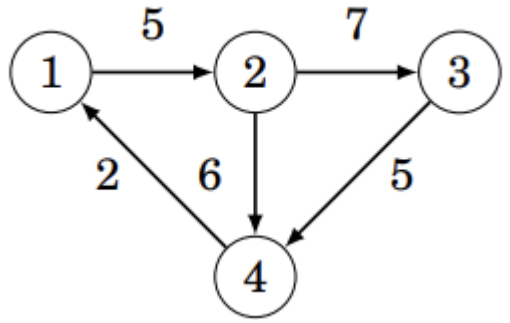


	1	2	3	4
1	0	5	0	0
2	0	0	7	6
3	0	0	0	5
4	2	0	0	0

Представление ребер (edge list representation) C++



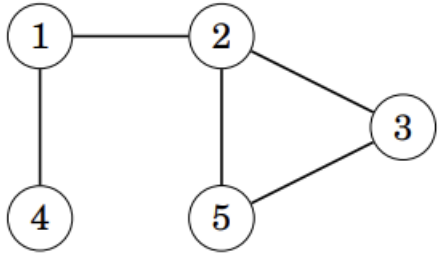

```
vector<pair<int,int>> edges;
edges.push_back({1,2});
edges.push_back({2,3});
edges.push_back({2,4});
edges.push_back({3,4});
edges.push_back({4,1});
```

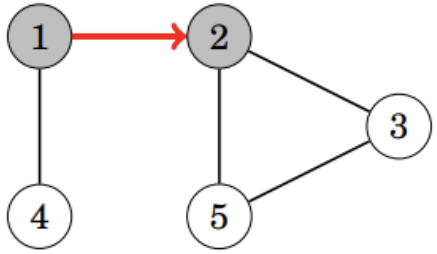


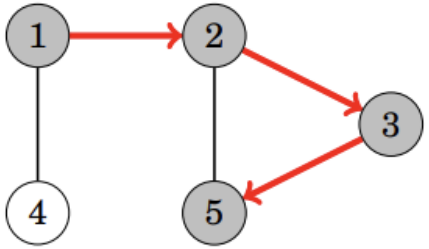

```
vector<tuple<int,int,int>> edges;
edges.push_back({1,2,5});
edges.push_back({2,3,7});
edges.push_back({2,4,6});
edges.push_back({3,4,5});
edges.push_back({4,1,2});
```

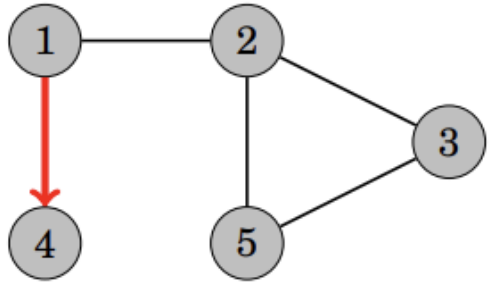
DFS

- Поиск в глубину (DFS) — это простой метод обхода графа. Алгоритм начинается с начального узла и переходит ко всем другим узлам, достижимым из начального узла с использованием ребер графа. Поиск в глубину всегда следует по одному пути в графе, пока он находит новые узлы. После этого он возвращается к предыдущим узлам и начинает исследовать другие части графа. Алгоритм отслеживает посещенные узлы, поэтому обрабатывает каждый узел только один раз.









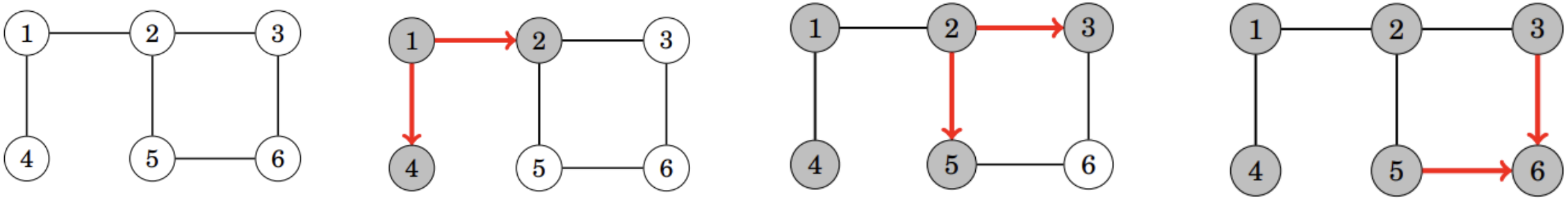
DFS C++

```
vector<int> adj[N];
Bool visited[N];

void dfs(int s) {
if (visited[s]) return;
visited[s] = true;
// process node s
for (auto u: adj[s]) {
 dfs(u);
 }
}
```

BFS

- Поиск в ширину (BFS) посещает узлы в порядке возрастания их расстояния от начального узла. Таким образом, мы можем вычислить расстояние от начального узла до всех остальных узлов, используя поиск в ширину. Однако поиск в ширину реализовать сложнее, чем поиск в глубину. Поиск в ширину проходит через узлы один уровень за другим. Сначала поиск исследует узлы, расстояние которых от начального узла равно 1, затем узлы, расстояние которых равно 2, и так далее. Этот процесс продолжается до тех пор, пока все узлы не



BFS C++

```
queue<int> q;
bool visited[N];
int distance[N];
visited[x] = true;
distance[x] = 0;
q.push(x);
while (!q.empty()) {
int s = q.front(); q.pop();
// process node s
for (auto u : adj[s]) {
if (visited[u]) continue;
visited[u] = true;
distance[u] = distance[s]+1;
q.push(u); } }
```

Выводы

1. Граф G = (V, E) представляют списком смежности или матрицей смежности — как ориентированный, так и неориентированный.
2. Список смежности предпочтителен для разреженных графов, матрица смежности — для плотных графов и быстрой проверки наличия ребра.
3. Сумма длин списков смежности равна 2|E| для неориентированного и |E| для ориентированного графа.
4. DFS идёт вглубь по достижимым вершинам; BFS посещает вершины в порядке возрастания расстояния от начальной.

Контрольные вопросы

1. Чем ориентированный граф отличается от неориентированного?
2. Как устроены список смежности и матрица смежности? Сколько памяти они занимают?
3. В каких случаях следует выбирать каждое из представлений?
4. Как проверить наличие ребра (u, v) в списке и в матрице смежности?
5. Опишите алгоритм поиска в глубину (DFS).
6. Опишите алгоритм поиска в ширину (BFS). Какая структура данных в нём используется?

Список литературы

Основная литература

1. Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ. 3-е изд. — М.: Вильямс, 2013.
2. Скиена С. Алгоритмы. Руководство по разработке. 2-е изд. — СПб.: БХВ-Петербург, 2011.
3. Седжвик Р., Уэйн К. Алгоритмы на Java. 4-е изд. — М.: Вильямс, 2013.

Дополнительная литература

4. Дасгупта С., Пападимитриу Х., Вазирани У. Алгоритмы. — М.: МЦНМО, 2014.
5. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. — Addison-Wesley, 1983.
6. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms. 4th ed. — MIT Press, 2022.

Электронные ресурсы

7. VisuAlgo — визуализация алгоритмов и структур данных: visualgo.net
8. Справочник по C++ и STL: en.cppreference.com

Спасибо за внимание!