Сравнительный анализ алгоритмов: Поиск в Ширину (BFS) и Глубину (DFS)

Николаев Дмитрий Иванович

Российский университет дружбы народов имени Патриса Лумумбы (РУДН)

1. Поиск в ширину (BFS)

Идея: ”Волна”. Алгоритм исследует граф слоями. Сначала посещаются все соседи текущей вершины, затем соседи соседей.

Применение: Поиск кратчайшего пути в невзвешенных графах, анализ социальных сетей (теория шести рукопожатий).

2. Алгоритм BFS

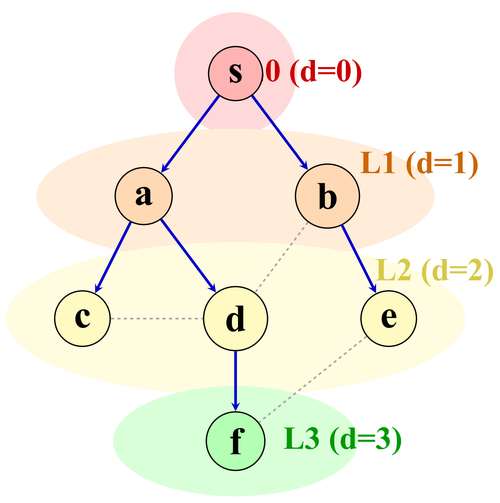
Алгоритм 1 BFS Iterative

Вход: Граф G, старт s

```
1:  Q.push(s); visited[s] ← true
2:  while Q не пуста do
3:      u ← Q.dequeue()
4:      for all v ∈ Adj[u] do
5:          if !visited[v] then
6:              visited[v] ← true
7:              d[v] ← d[u] + 1
8:              Q.enqueue(v)
9:          end if
10:     end for
11: end while
```

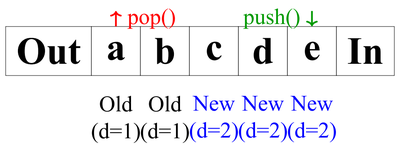
3. Визуализация BFS (Слои)

Вершины сгруппированы по расстоянию от s. Фон показывает ”волны” поиска.



4. Структура данных: Очередь

Состояние очереди Q в момент, когда слой L1 (a, b) обработан, а L2 добавляется.



FIFO (First-In First-Out): Гарантирует, что вершины с $d = k$ выйдут раньше, чем вершины с $d = k + 1$.

5. Сравнение BFS и DFS

	BFS (Ширина)	**DFS (Глубина)**
Структура	Queue (Очередь)	Stack (Стек)
Порядок	Слой за слоем	Вглубь до упора
Путь	**Кратчайший**	Случайный (длинный)
Память	$O(\|V\|)$ (ширина)	$O(h)$ (глубина)
Сложность	$O(V + E)$	$O(V + E)$

6. Математическая база

Теорема (Корректность BFS): Пусть $\delta(s, v)$ — кратчайшее расстояние от s до v. BFS вычисляет значения $d[v]$ так, что $d[v] = \delta(s, v)$ для всех $v \in V$.

Доказательство (через инвариант): Значения d в очереди не убывают: если $Q = \langle v_1, \ldots, v_k \rangle$, то $d[v_1] \leq d[v_2] \leq \cdots \leq d[v_k]$ и $d[v_k] \leq d[v_1] + 1$. При извлечении u и добавлении соседа v, мы ставим $d[v] = d[u] + 1$, сохраняя порядок слоев.

В BFS очередь хранит ”активный фронт” волны!

7. Поиск в глубину (DFS)

Идея: ”Лабиринт”. Идем по ребру как можно дальше. Если зашли в тупик — возвращаемся назад (backtrack).

Применение: Топологическая сортировка, поиск циклов, компонент сильной связности, проверка на двудольность.

8. Алгоритм DFS

Алгоритм 2 DFS Recursive

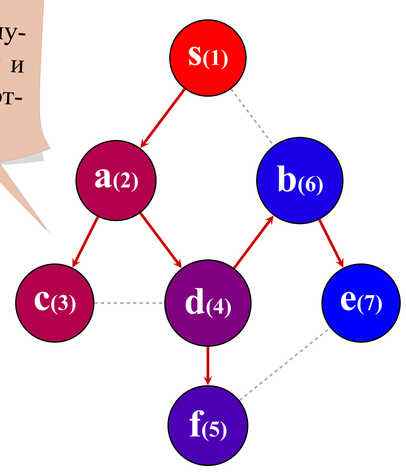
Вход: Граф G, вершина u

```
1: visited[u] ← true
2: for all v ∈ Adj[u] do
3:     if !visited[v] then
4:         π[v] ← u
5:         DFS(G, v)
6:     end if
7: end for
```

9. Визуализация DFS (Вглубь)

Тот же граф, но порядок обхода другой. Цвет показывает время посещения ($1 \to 7$).

Дерево DFS получается глубоким и извилистым, в отличие от BFS.



Легенда: Цифры (1..7) — порядок посещения. Красные стрелки — путь рекурсии.

10. Глубокий анализ: Алгебра и Топология

1. Матричная интерпретация BFS

Пусть A — матрица смежности графа. Количество путей длины k между вершинами i и j равно $(A^k)_{ij}$. Расстояние в BFS можно определить через степени матрицы в булевой алгебре (или $(+, \times) \to (\min, +)$ полукольце):

$$d(s, v) = \min\{k \in \mathbb{N} \mid (A^k)_{sv} > 0\}$$

Это свойство используется в алгоритмах на GPGPU (параллельные вычисления), где матричное умножение эффективно распараллеливается.

2. Теорема о скобочной структуре DFS

Для любых двух вершин u и v отрезки времени их обработки $[entry[u], exit[u]]$ и $[entry[v], exit[v]]$ либо не пересекаются, либо один строго вложен в другой.

Классификация ребер (u, v) при DFS:

- **Tree Edge:** v открыта из u (белая).
- **Back Edge:** v — предок u (серая). $\implies$ Цикл!
- **Forward Edge:** v — потомок u (черная).
- **Cross Edge:** v не предок и не потомок (черная).

Резюме

Понимание разницы между BFS и DFS критически важно для Computer Science. BFS идеален для поиска кратчайших путей и работы со слоистыми структурами. DFS незаменим, когда нужно исследовать всё пространство состояний или найти сложные связности (циклы, мосты). Оба алгоритма имеют линейную сложность, но совершенно разную философию обхода.