# Алгоритм Хопкрофта-Карпа

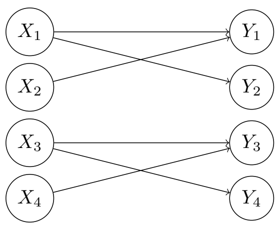
*Подготовил Сивухин Никита. По вопросам пишите на почту sivukhin.work+teach@gmail.com*

Вспомним как выглядит оптимизированная реализация алгоритма Куна:

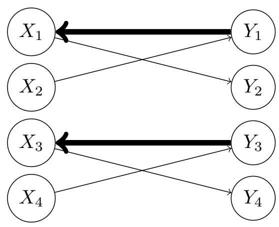
```cpp
struct Graph {
  vector<int> match() {
    auto match = vector<int>(m, -1);
    auto used = vector<int>(n, 0);
    bool increased = false;
    do {
        // you can reset colors in O(1) if you will maintain "current color" instead
        fill(used.begin(), used.end(), 0);
        increased = false;
        for (int v = 0; v < n; v++) {
            increased |= match[v] == -1 && try_kuhn(v, match, used);
        }
    } while (increased);
    return match;
  }
};
```

Листинг 1: Алгоритм Куна

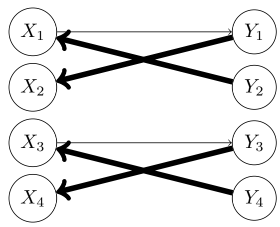
Так как пометки посещённых вершин обновляются только перед очередной итерацией внешнего цикла, то в данном виде алгоритм Куна находит максимальное по включению множество вершинно-непересекающихся $M$-чередующихся цепей за время $O(n+m)$.




Изначальный граф




Первая итерация алгоритма 1




Вторая итерация алгоритма 1

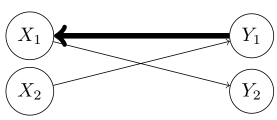
Основная идея алгоритма Хопкрофта-Карпа заключается также в том, чтобы искать множество вершинно-непересекающихся $M$-чередующихся цепей за один обход графа но так, чтобы после чередования всех найденных цепей любая новая $M$-чередующаяся цепь была строго длиннее чем все найденные ранее.

Если добиться выполнения данного свойства, то несложно показать что количество шагов алгоритма не превосходит $O(\sqrt{n})$. Действительно, заметим что после $\sqrt{n}$ итераций $M$-чередующиеся цепи относительно текущего паросочетания должны иметь длину хотя бы $\sqrt{n}$. Теперь, если посмотреть на симметрическую разность $M \oplus M_{opt}$, то в данном графе чередующиеся относительно $M$ цепи нечётной длины имеют длину хотя бы $\sqrt{n}$, а значит их не больше чем $\sqrt{n}$ и $|M_{opt}| - |M| \le \sqrt{n}$.

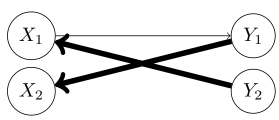
Теперь перейдем к формальному описанию алгоритма и фактов, на которых он базируется.

**Лемма 1.** *Если $P$ — кратчайшая $M$-чередующаяся цепь, то любая $M'$-чередующаяся цепь $P'$ для паросочетания $M' = M \oplus P$ имеет длину $|P'| \ge |P| + 2|P \cap P'|$.*

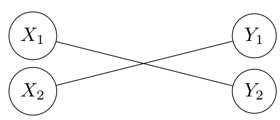
*Доказательство.* Рассмотрим паросочетание $N = M \oplus P \oplus P'$. Так как $P$ и $P'$ — чередующиеся цепи, то $|N| = |M|+2$, а значит в графе $M \oplus N = P \oplus P'$ существует две цепи $P_1$ и $P_2$ чередующиеся относительно $M$. Так как $P$ — кратчайшая $M$-чередующаяся цепь, то $|P_1| \ge |P|$ и $|P_2| \ge |P|$, а значит $|P \oplus P'| \ge |P_1| + |P_2| \ge 2|P|$. Однако также размер симметрической разности путей выражается как $|P \oplus P'| = |P| + |P'| - 2|P \cap P'|$, а значит $|P| + |P'| - 2|P \cap P'| \ge 2|P| \Rightarrow |P'| \ge |P| + 2|P \cap P'|$ □




$M \oplus P, |P| = 1$



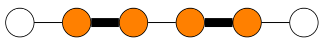

$M \oplus P \oplus P', |P'| = 3$



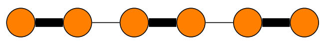

$P \oplus P', |P \oplus P'| \ge 2|P|$

**Лемма 2.** *Если $P$ — $M$-чередующаяся цепь, то любая $M'$-чередующаяся цепь $P'$ для паросочетания $M' = M \oplus P$ либо имеет общее ребро с $P$, либо вершинно не пересекается с $P$.*

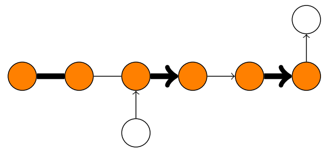
*Доказательство.* Заметим, что все вершины из $P$ в паросочетании $M \oplus P$ насыщены, а это значит, что они являются внутренними вершинами пути $P'$ и им инцидентно ровно одно ребро из паросочетания в $M \oplus P$. Значит, у $P$ и $P'$ есть общая вершиа $v$, то ребро паросочетания $M \oplus P$ инцидентное данной вершине также принадлжит обеим цепям. $\square$



$M$-чередующийся путь $P$



Путь $P$ в $M \oplus P$



Возможная цепь $P'$ в $M \oplus P$

Теперь, будем итеративно строить максимальные по включению множества $S_i$ кратчайших $M$-чередующихся цепей относительно текущего паросочетания $M$.

Более формально, начиная с пустого паросочетания $M_1 = \emptyset$, на каждой итерации будем строить **максимальное по включению** множество **вершинно-непересекающихся** чередующихся цепей $S_i = \{P_{i,1}, P_{i,2}, \ldots, P_{i,k_i}\}$ имеющих **кратчающую длину** относительно паросочетания $M_i = M_{i-1} \oplus \bigoplus_{P \in S_{i-1}} P$.

Заметим, что так как все пути в $S_i$ имеют кратчающую длину (т.е. $|P_{i,a}| = |P_{i,b}| = \ell_i$), а множество $S_i$ максимальное по включению, то из лемм 1 и 2 следует, что для любого пути из множества $S_{i+1}$ верно, что $|P_{i+1}, a| > \ell_i$, а значит $\ell_i \geq i$ для любого $i$.

Теперь рассмотрим произвольное наибольшее паросочетание $M_{opt}$ и построим его симметрическую разность $H$ с паросочетанием $M_{mid} = M_{\lceil\sqrt{n}\rceil}$. Заметим, что в $H$ все чередующиеся цепи относительно $M_{mid}$ имеют длину хотя бы $\sqrt{n}$, т.к. $\ell_{\lceil\sqrt{n}\rceil} \geq \sqrt{n}$, а значит $|M_{opt}| - |M_{mid}| \leq \sqrt{n}$ и количество множеств $S_i$ не превосходит $2\lceil\sqrt{n}\rceil = O(\sqrt{n})$.

Можно также доказать более строгую оценки с помощью следующего утверждения:

**Лемма 3.** *Если наибольшее паросочетание $M_{opt}$ имеет размер $s$, то для любого паросочетания $M$ размера $r$ верно, что в графе существует $M$-чередующаяся цепь длиной не более $2\lceil\frac{r}{s-r}\rceil + 1$.*

*Доказательство.* Рассмотрим граф $H = M_{opt} \oplus M$. В данном графе существует хотя бы $s - r$ цепей, являющихся чередующимися отсносительно $M$. Рассмотрим цепь $P$ с минимальной длиной среди таких. В данной цепи не может быть более $\lceil\frac{r}{s-r}\rceil$ рёбер из $M$, так как она минимальная. А значит длина $|P|$ не превосходит $2\lceil\frac{r}{s-r}\rceil + 1$. $\square$

Тогда, если $s = |M_{opt}|$, то рассмотрим паросочетание $M_{mid}$ размера $r = |M_{mid}|$, построенное к шагу $mid = \lceil\sqrt{s}\rceil$. К данному шагу чередующиеся цепи имеют длину не менее чем $\sqrt{s}$, а значит $2\lceil\frac{r}{s-r}\rceil + 1 \geq \sqrt{s}$, откуда следует, что $r \geq s - O(\sqrt{s})$, а значит всего шагов алгоритма не более чем $O(\sqrt{s})$.

**Замечание 1.** *Заметим, что все доказательства приведённые выше применимы к любому графу. Двудольность в алгоритме Хопкрофта-Карпа необходима чтобы эффективно искать множества $S_i$ на каждом шагу (также как и с теоремой Бержа и алгоритмом Куна)*

**Упражнение 1.** *Постройте алгоритм нахождения $(1 + \frac{1}{\varepsilon})$-апроксимации наибольшего паросочетания в двудольном графе, работающий за время $O(\varepsilon(n + m))$*

## Детали реализации

Реализации каждого шага алгоритма Хопкрофта-Крафта состоит из двух этапов:

1. Сначала нужно найти длину кратчайшего $M_i$-чередующегося пути (`min_distance`) и построить граф кратчайших путей (`layers`), путём запуска поиска в ширину по графу $G$ с наведённой ориентацией относительно текущего $M_i$

2. После этого нужно запустить поиск в глубину, чтобы выбрать максимальное по включению множество вершинно-непересекающихся кратчайших путей, пользуясь графом кратчайших путей и длиной кратчайшего пути, найденными на прошлом этапе

Набросок реализации представлен в листинге 2 (заметьте, что алгоритм сильно похож на оптимизированную версию алгоритма Куна 1).

```
struct Graph {
  bool try_hc(vector<int> &match, int d, vector<int> &layers, vector<int> &used, int v) {
    if (used[v] || d < 0) {
      return false;
    }
    used[v] = 1;
    for (int to : edges[v]) {
      if (match[to] == -1 ||
          layers[match[to]] == layers[v]+2 && try_hc(match[to], match, d-2, layers, used)) {
        match[to] = v;
        return true;
      }
    }
    return false;
  }
  vector<int> match() {
    auto match = vector<int>(m, -1);
    auto used = vector<int>(n, 0);
    auto layers = vector<int>(n, 0);
    bool increased = false;
    do {
        // run BFS from "free" verices of left part
        // and calculate min distance of agumenting path
        fill(used.begin(), used.end(), 0);
        fill(layers.begin(), layers.end(), 0);
        auto min_distance = bfs(match, layers);

        // run DFS to find "blocking" set of shorted augmenting paths
        fill(used.begin(), used.end(), 0);
        increased = false;
        for (int v = 0; v < n; i++) {
            increased |= match[v] == -1 && try_hc(v, match, min_distance, layers, used);
        }
    } while (increased);
    return match;
  }
};
```

Листинг 2: Алгоритм Хопкрофта-Карпа

## Ссылки


- An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs: https://ieeexplore.ieee.org
- Алгоритм Хопкрофта-Карпа: https://ru.wikipedia.org/wiki
- Bipartite Matching in Nearly-linear Time on Moderately Dense Graphs: https://arxiv.org/abs/2009.01802