50. Алгоритм A^*, определение функций f, g, h; реализация

Применение: нахождение минимального пути от вершины s до вершины t в графе с неотрицательными рёбрами.

A^* работает по принципу алгоритма Дейкстры.
$g(v)$ – текущее кратчайшее расстояние от s до v (из алгоритма Дейкстры)

$h(v)$ – некоторая оценка на $dist(v, t)$. ($h(v)$ называют *эвристикой*)
$f(v) = g(v) + h(v)$

Смысл алгоритма: следующую вершину выбираем не по $g(v)$ (как в алгоритме Дейкстры), а по $f(v)$.

Реализация (псевдокод):

q – очередь, сравнение элементов по функции f. $g[s] = 0$ – задали базу для функции g
gr – граф, заданный списком рёбер.

```

q.insert(s); // добавляем стартовую вершину

while(!q.empty()) {
v = q.top();
q.pop();

for (edge e : gr[v]) {
    c = g[v] + e.cost + h(to);
    if (c < f[to]) {
        g[e.to] = g[v] + e.cost;
        if (to ∈ q) {
            q.decreaseKey(...); // уменьшить у вершины e.to значение f до g[e.to] + h(e.to)
        } else {
            q.insert(...); // добавить вершину e.to со значением f = g[e.to] + h(e.to)
        }
    }
}
```

51. Вырожденные случаи в алгоритме A^*: $h \equiv 0$, $h(v) = dist(v, t)$

1) $h \equiv 0$. Получается, что сравнение идёт только по $g(v)$, а значит алгоритм A^* вырождается в алгоритм Дейкстры.

2) $h(v) = dist(v, t)$ – эвристика всегда каким-то образом знает точное расстояние от v до t.
Тогда A^* рассматривает почти только оптимальный путь (почти только означает, что алгоритм рассматривает не только вершины, принадлежащие кратчайшему пути, но и их соседей).

Доказательство. Докажем сначала, что если h – монотонна, то значения f в куче не убывают (у извлекаемых элементов).

Пусть мы раскрываем вершину v. Рассмотрим ребро (v, u).
$g(u) = g(v) + cost(v, u)$
С другой стороны, так как h – монотонна, то $h(v) \le h(u) + cost(v, u)$, то есть $h(u) \ge h(v) - cost(v, u)$. Сложим равенство и неравенство, получим:

$$g(u) + h(u) = f(u) \ge g(v) + h(v) = f(v)$$

Доказано.
Тогда получается, что каждая вершина раскроется не более одного раза. (Предположим противное, тогда получаем, что во второй раз мы её раскрыли со значением f меньшим, чем в первый раз. Противоречие.

■

Наша эвристика такая, что значения f одинаковы для всех вершин, а значит каждая вершина раскроется не более одного раза. По этому утверждению и принципу работы алгоритма понятно, что он рассматривается почти только оптимальный путь.

52. Допустимые и монотонные эвристики в алгоритме A^*. Примеры монотонных эвристик на разных сетках.

Определение. Эвристика $h(v)$ называется *допустимой*, если $\forall v \; h(v) \le dist(v, t)$.

Следствие. Если $h(v)$ – допустимая эвристика, то $h(t) = 0$.

Замечание (Дополнительно). *Если эвристика $h(v)$ – недопустимая, то A^* находит неточный ответ, на практике с помощью недопустимых эвристик можно добиться, чтобы алгоритм работал быстро, при этом ответ не сильно отличался от правильного.*

Определение. Эвристика $h(v)$ называется *монотонной*, если:
1) $h(t) = 0$ 2) $\forall (u, v) \in E \; h(u) \le h(v) + cost(u, v)$, где E – множество рёбер (неравенство треугольника).

Следствие. Монотонная эвристика является допустимой.

Примеры эвристик

1) Если граф представляет собой неполную сетку (то есть из каждой точки потенциально есть 4 ребра: вверх, вниз, вправо, влево), при этом некоротых рёбер может не быть (то есть не у всех точек есть ровно 4 ребра), то в качестве эвристики можно использовать Манхэттенское расстояние: $h(v) = |v.x - t.x| + |v.y - t.y|$ (каждую вершину можно задать двумя координатами на плоскости).

2) Если в графе также можно ходить по диагонали, то в качестве эвристики можно использовать расстояние Чебышёва: $h(v) = max\{|v.x - t.x|, |v.y = t.y|\}$

3) Если можно ходить по плоскости куда угодно, то в качестве эвристики можно использовать Евклидово расстояние: $h(v) = \sqrt{(v.x - t.x)^2 + (v.y - t.y)^2}$.

53. Формулировка работоспособности (корректность и время работы) алгоритма A^* в случае монотонной, допустимой или произвольной эвристики. Доказательство для монотонного случая.

В случае произвольной эвристики A^* может довольно быстро найти хорошее приближение.

В случае монотонной и допустимой эвристики алгоритм находит точный минимальный путь, при этом в случае монотонной эвристики алгоритм раскрывает каждую вершину не более 1 раза, в случае допустимой эвристики может работать довольно долго.

Доказательство (для монотонной эвристики). Пусть h – монотонная эвристика. Алгоритм A^* на каждом шаге раскрывает вершину с минимальным значением f, при этом. Когда алгоритм дойдет до вершины t, он извлечет её из кучи со значением $f(t)$, но $f(t) = g(t) + h(t) = g(t)$ ($h(t) = 0$ по определению монотонной эвристики). Осталось доказать, что $g(t)$ – не только оценка сверху для минимального пути, но и в точности равна ей. Пусть оптимальный путь $OPT < f(t)$. Тогда бы он извлёкся из кучи раньше, так как значения f у извлекаемых элементов не убывают (доказано в пункте 51). Противоречие. ■

54. Алгоритм Флойда: поиск попарных кратчайших расстояний в графе без отрицательных циклов. Реализация, асимптотика.

Применение: поиск попарных кратчайших расстояний.

Требования к графу: допускаются отрицательные рёбра, но нет отрицательных циклов.

Описание алгоритма:

Введём трехмерную динамику:
$dp[i][j][k]$ – минимальная длина пути между i и j, такая, что все промежуточные вершины имеют номера не больше, чем k.

База:
$dp[i][j][0] =$ вес ребра из i в j ($+\infty$, если ребра нет)

Переход:
Пусть посчитаны первые k слоёв. Построим $k + 1$ слой.
$dp[i][j][k + 1] = min(dp[i][j][k], dp[i][k + 1][k] + dp[k + 1][j][k])$, то есть либо не берём $k + 1$ вершину, либо берем.

Ответ: $dp[i][j][n]$, где n – количество вершин.

Асимптотика: $O(n^3)$
Память: $O(n^3)$

Память можно оптимизировать до $O(n^2)$, если в качестве dp использовать матрицу смежности графа:

Пусть g – матрица смежности графа. Тогда алгоритм Флойда можно написать так:

```
for (k = 1 ... n) {
    for (i = 1 ... n) {
        for (j = 1 ... n) {
            g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
        }
    }
}

```

55. Восстановление ответа (пути) в алгоритме Флойда.

Заведем массив $p[i][j]$, заполним его каким-то нейтральным элементом (например, -1). Если на каком-то шаге выгоднее идти через вершину k, то есть $dp[i][j][k+1] == dp[i][k+1][k] + dp[k+1][j][k]$, то сохраняем это: $p[i][j] = k$.

Теперь найти путь между i и j можно рекурсивно:
Если на какой-то этапе $dp[i][j] == -1$, то кратчайший путь – ребро между i и j.
Если $dp[i][j] = k \neq -1$, то путь из i в j – это путь из i в k и из k в j (их находим рекурсивно).

56. Алгоритм Форда—Беллмана: поиск кратчайших расстояний от одной вершины до всех. Реализация, асимптотика (в случае отсутствия отрицательных циклов).

Применение: нахождение минимального пути от вершины s до вершины t (в графе могут быть и отрицательные рёбра, и отрицательные циклы).

Описание алгоритма:

Введём двумерную динамику:
$dp[v][k]$ – кратчайшее расстояние от s до v при использовании не более, чем k рёбер.

База: $dp[v][0] = \begin{cases} 0, v = s \\ +\infty, v \neq s \end{cases}$

Переход:
Пусть посчитаны первые k слоёв. Тогда $dp[u][k+1] = \min(dp[u][k], \min\limits_{(v,u)} dp[v][k] + cost(v,u))$, то есть либо оставляем ответ, достижимый за k рёбер, либо перебираем все входящие в u рёбра, и выбираем из них минимальное.

Ответ:
dp[v][n - 1], если нет отрицательных циклов

Асимптотика: $O(nm)$, n – количество слоёв в dp, m – переход между слоями.

57. Алгоритм Форда—Беллмана: нахождение кратчайших расстояний от одной вершины до всех в случае наличия отрицательных циклов.

Для обработки отрицательных циклов найдём дополнительно n-ый слой $dp[v][n]$.

Утверждение. *Пусть C – отрицательный цикл, достижимый из s. Тогда $\exists v \in C : dp[v][n] < dp[v][n-1]$.*

Доказательство. Пусть $c_1, ..., c_k$ – веса рёбер в цикле C. Предположим противное: $\forall v \in C dp[v][n] = dp[v][n-1]$ (больше оно быть не может по переходу динамики).
Пусть c_i – вес ребра, соединяющего вершины v_i и v_{i+1}. Тогда $dp[v_{i+1}][n] \leq dp[v_i][n-1] + c_i$. Сложим эти неравенства по всем i.

$$\sum_{v_i \in C} dp[v_i][n] \leq \sum_{v_i \in C} do[v_i][n-1] + \sum_i c_i$$

Первая и вторая суммы равны по предположению, третья сумма меньше нуля (так как цикл отрицательный). Получается, что 0 меньше или равен чего-то отрицательного. Противоречие. ∎

Тогда за $O(nm)$ можем найти хотя бы по одной вершине на каждом отрицательном цикле.

$$dist(s, x) = \begin{cases} -\infty, \text{если } x \text{ достижим из одной из вершин отрицательного цикла} \\ dp[x][n-1], \text{иначе} \end{cases}$$

На практике лучше запустить DFS из всех вершин, у которых $dp[v][n] < dp[v][n-1]$, и пометить все достижимые из них вершины $dist = -\infty$.