A*

Eduard Batmendijn

5. marca 2016

Abstrakt

Na hľadanie najkratších ciest poznáme Dijkstrov algoritmus, ktorý funguje vcelku rýchlo na ľubovoľnom grafe (s nezápornými hranami). Niekedy je však graf taký veľký, že aj Dijkstra je príliš pomalý. Na oplátku však o grafe môžeme nejakú informáciu navyše. Algoritmus A*, o ktorom je táto prednáška, je zovšeobecnenie Dijkstrovho algoritmu, ktoré dokáže z informácie navyše, ktorú mu dodáme, vytrieskať nejakú tú rýchlosť navyše.

Spomínanú "informáciu navyše" dávame algoritmu A* vo forme heuristiky, ktorá odhaduje vzdialenosti vrcholov od cieľa.

Definícia 1. *Heuristikou* budeme rozumieť funkciu h, ktorá ľubovoľnému vrcholu x nášho grafu priradí nejaký odhad jeho vzdialenosti od cieľa.

Definícia 2. Heuristika h je

- *prípustná (optimistická)*, ak žiadnemu vrcholu nepriradí vzdialenosť väčšiu, než je jeho skutočná vzdialenosť od cieľa.
- *monotónna (konzistentná)*, ak spĺňa trojuholníkovú nerovnosť, t. j. $h(x) \leqslant d(x, y) + h(y)$ pre každú hranu x, y, kde $d(x, y)$ je dĺžka tejto hrany.

Na to, aby A* fungoval, potrebujeme, aby naša heuristika bola zároveň prípustná aj monotónna.

Listing programu (C++)

```
double h(int a, int b);

struct edge
{
        int a, b;
        double length;

        int ten_druhy(int ja)
        {
                return a^b^ja;
        }
};


vector<int> a_star(vector<vector<int> >& susedna, vector<edge>& hrana, int odkial, int kam)
{
        int n = susedna.size();
        vector<double> dist(n, inf);
        vector<int> from(n, -1);

        dist[odkial] = 0;
        priority_queue<pair<double, int>, vector<pair<double, int> >, greater<pair<double, int> > > halda;
        halda.push(pair<double, int> (0, odkial));
        while(!halda.empty())
        {
                pair<double, int> cur = halda.top();
                halda.pop();
                int ja = cur.second;
                if(cur.first > dist[ja])continue;
                if(ja == kam)break;
                for(int i=0; i<susedna[ja].size(); i++)
                {
                        int h_id = susedna[ja][i];
                        int on = hrana[h_id].ten_druhy(ja);
                        if(dist[on] > dist[ja] + hrana[h_id].length + h(on, kam) - h(ja, kam))
                        {
                                dist[on] = dist[ja] + hrana[h_id].length + h(on, kam) - h(ja, kam);
                                from[on] = ja;
                                halda.push(pair<double, int> (dist[on], on));
                        }
                }
        }
        vector<int> path;
        for(int cur = kam; cur != -1; cur = from[cur])
        {
                path.push_back(cur);
        }
        reverse(path.begin(), path.end());
        return path;
}
```