Task Discussion
Odd Route

Andreas Bärtschi
based on slides by Jan Hązła

ETH Zürich

December 16, 2015

The problem

Given a directed weighted graph G with n vertices and m edges.

Find a shortest s-t path such that both its number of edges and total weight are odd.

$\boldsymbol{n \leq 10^5,\ m < 2 \cdot 10^5.}$
$n \approx m$.

Acceptable running time for the algorithm:

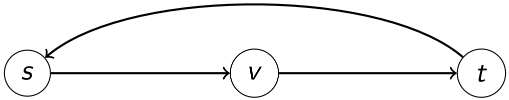
- $n + m$,
- $m \cdot \log m$,

$\Rightarrow$ linear $\cdot$ polylogarithmic combinations of n and m.

Solution – simplified case

Hint: Assume all the weights are 1.
In this case we can look for the shortest path with odd number of edges.

Note that the path does not have to be simple:


$$\begin{tikzcd} s \arrow[r] & v \arrow[r] & t \arrow[ll, bend right] \end{tikzcd}$$

Recall **Tracking** Task:
A shortest journey between cities x and y such that at least k edges go along a river.
A shortest path between nodes x and y such that at least k edges have given property.

Solution – the trick

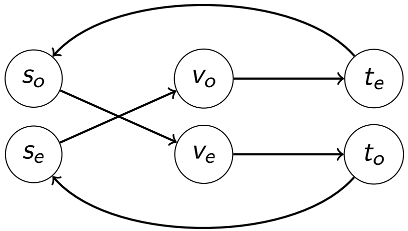
Leveling as in Tracking not possible here (we go back and forth between odd/even number of edges instead of having a monotonically growing k).

DP-like approach: For each vertex v differentiate between
being in v after an *even* number of steps and
being in v after an *odd* number of steps.

Define H as follows:
For each $v \in V(G)$ put v_o and v_e in $V(H)$.

For each $e = (u, v)$ in $E(G)$ put
(u_o, v_e) and (u_e, v_o) in $E(H)$.

Output **shortest path from s_e to t_o.**



Solution – full problem

With arbitrary weights we also have to care about whether the **weight up to the current vertex** is **odd or even**.
Use the previous idea with four vertices per $v \in V(G)$: $\boldsymbol{v_{ee}}$, $\boldsymbol{v_{eo}}$, $\boldsymbol{v_{oe}}$ and $\boldsymbol{v_{oo}}$.
Proof: Exercise.

Implementation:

- Construct a graph with $4n$ vertices and $4m$ edges and call BGL Dijkstra.
  Complexity: $O(m \log m)$. Lines of code: ~ 50.
- A vertex from G in the range $v \in \{0, \dots, n-1\}$
  can be represented in H by $\{4v, \dots, 4v+3\}$.
- Partial points for DAG case can be obtained with a DP solution
  (but with basically the same idea).

Code snippet – edge insertion

```
// Split all vertices u, 0 <= u < V(G) into 4 parts
// 4*u   (even length, even weight)
// 4*u+1 (even length, odd weight)
// 4*u+2 (odd length, even weight)
// 4*u+3 (odd length, odd weight)
Graph G(4*n);
WeightMap weight = get(edge_weight, G);
// add (u -> v) from E(G) to H}
Edge e;
tie(e, tuples::ignore) = add_edge(4*u,   4*v+2+w%2, H); weight[e] = w;
tie(e, tuples::ignore) = add_edge(4*u+1, 4*v+3-w%2, H); weight[e] = w;
tie(e, tuples::ignore) = add_edge(4*u+2, 4*v+w%2,   H); weight[e] = w;
tie(e, tuples::ignore) = add_edge(4*u+3, 4*v+1-w%2, H); weight[e] = w;
```