Problem L: Well Spoken

Algorithmen für Programmierwettbewerbe

Hendrik Meininger          Johannes Schleicher
Problem

Given:
- Timespan [A:B]
- Road network
- Intersections as nodes
- Streets as edges

Problem:
Find the minimum maximum waiting time, given that Janet be ready between [A:B]
Problem L: Well Spoken

Hendrik Meininger, Johannes Schleicher

Input:

<table>
<thead>
<tr>
<th>[A:B]</th>
<th>10 20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3 5</td>
</tr>
<tr>
<td></td>
<td>1 3 7</td>
</tr>
<tr>
<td></td>
<td>2 1 1</td>
</tr>
<tr>
<td></td>
<td>2 3 2</td>
</tr>
<tr>
<td></td>
<td>2 3 5</td>
</tr>
<tr>
<td></td>
<td>3 2 4</td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>[10:20]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output is always an int</td>
</tr>
</tbody>
</table>

Output:

6

[Diagram of a network with nodes 1, 2, and 3, and edges with costs 1, 2, 5, and 4.]
### Problem:
Find the minimum maximum waiting time, given that Janet be ready between [A:B]

### Solution:
1. Compute the distance from Richard to all vertices and from all vertices to Janet using two runs of Dijkstra
2. Binary search on the maximum waiting time boundaries
3. Check if given delay $\delta$ is possible
Problem L: Well Spoken

Approach - Example

Problem Approach Runtime

\[ 0 \leq \text{delay} \leq 7 \rightarrow \delta = \left\lceil \frac{7 - 0}{2} \right\rceil + 0 = 3 \]

\[ [A: B] = [10: 20] \]
## Approach

### Problem L: Well Spoken

### Approach

Check if delay $\delta$ is possible:

1. Mark vertices $u$ as "good" if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$
   
   If node $u$ does not satisfy this condition, then there won’t be a route through $u$ that satisfies the delay delta, given that the signal will come at time $A$ or when arriving at $u$
1. Step:
$0 \leq delay \leq 7 \rightarrow \delta = 3$

Mark vertices $u$ as “good“ if:
$\text{distFromHome}[u] + \text{distToJanet}[u] \leq A + \delta$ and $\text{distToJanet}[u] \leq \delta$

$u_1$: $0 + 7 \leq 10 + 3$ and $7 \leq 3$
$u_2$: $11 + 2 \leq 10 + 3$ and $2 \leq 3$
$u_3$: $7 + 0 \leq 10 + 3$ and $0 \leq 3$
Approach

Check if delay $\delta$ is possible:

1. Mark vertices $u$ as “good“ if $distFromHome(u) + distToJanet(u) \leq A + \delta$ and $distToJanet(u) \leq \delta$
   
   If node $u$ does not satisfy this condition, then there won’t be a route through $u$ that satisfies the delay delta, given that the signal will come at time $A$ or when arriving at $u$.

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + distToJanet(v) \leq \delta$, then mark $v$ and edge $l$ as good too.
   
   An edge is good, should Richard still be able to meet the delay, if Janet calls while Richard is currently riding that edge.
1. Step:
0 ≤ delay ≤ 7 → δ = 3

Propagate: if u is good and u → v with l + distToJanet[v] ≤ δ,
then mark v and edge l as good too

u₂, e(2,1,1): 1 + 7 ≤ 3
u₂, e(2,3,2): 2 + 0 ≤ 3
u₂, e(2,3,5): 5 + 0 ≤ 3
u₃, e(3,2,4): 4 + 2 ≤ 3
Approach

Check if delay $\delta$ is possible:

1. Mark vertices $u$ as “good“ if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$
   If node $u$ does not satisfy this condition, then there won’t be a route through $u$ that satisfies the delay delta, given that the signal will come at time $A$ or when arriving at $u$

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too
   An edge is good, should Richard still be able to meet the delay, if Janet calls while Richard is currently riding that edge

3. If subgraph of good edges has cycle $\rightarrow$ delay $\delta$ is possible
   We can stay in the cycle until Janet calls and arrive at her place at most delta after she has called
1. Step:
\[0 \leq delay \leq 7 \rightarrow \delta = 3\]

If subgraph of good edges has cycle → delay \(\delta\) is possible
\(\text{hasCycle}() = false\)
### Approach

**Problem L: Well Spoken**

**Check if delay $\delta$ is possible:**

1. **Mark vertices $u$ as “good“ if** $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$
   
   If node $u$ does not satisfy this condition, then there won’t be a route through $u$ that satisfies the delay delta, given that the signal will come at time $A$ or when arriving at $u$.

2. **Propagate:** if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too.
   
   An edge is good, should Richard still be able to meet the delay, if Janet calls while Richard is currently riding that edge.

3. **If subgraph of good edges has cycle → delay $\delta$ is possible**
   
   We can stay in the cycle until Janet calls and arrive at her place at most delta after she has called.

4. **Otherwise, the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible.**
Approach – `longestPath()`

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approach</th>
<th>Runtime</th>
</tr>
</thead>
</table>

Auxiliary variables:

\[
indeg[u] := \text{indegree of good nodes in subgraph}
\]

- corresponds to the number of good incoming edges

\[
goodNodes := \text{u. isGood and indeg[u] = 0}
\]

- goodNodes contains at the beginning all good nodes with indegree 0, so that one can calculate the longest Path correctly afterwards

\[
latest[u] = A + \text{delay} - \text{distToJanet[u]}
\]

- for each good node \( u \), the latest time of arrival at the node, so that the delay can still be met
1. Step:
0 ≤ delay ≤ 7 → δ = 3

Compute longest time Richard can stay in the subgraph
If this is ≥ B then delay δ is possible

\[\text{latest}[u] = A + \text{delay} - \text{distToJanet}[u]\]

\[\text{latest}[2] = 10 + 3 - 2 = 11\]
\[\text{latest}[3] = 10 + 3 - 0 = 13\]
Pseudocode:

longestPath(δ)

while(! goodNodes.isEmpty())

    u = goodNodes.remove(0)

    if latest[u] + distToJanet[u] ≥ B
        return true

    for Edge e in goodEdges[u]

        indeg[e.dest] += 1

        if indeg[e.dest] == 0
            goodNodes.add(e.dest)

        latest[e.dest] = max(latest[e.dest], latest[u] + e.weight)

return false
1. Step:
0 ≤ delay ≤ 7 → δ = 3

Compute longest time Richard can stay in the subgraph
If this is ≥ B then delay δ is possible

\[ latest[u] = A + delay - distToJanet[u] \]

\[ latest[2] = 11 \]
\[ latest[3] = 13 \]

Start with good nodes with \( \text{indeg}[u] == 0 \)
Check if: \( latest[u] + distToJanet[u] ≥ B \)

\[ latest[3] = \max(latest[3], latest[2] + e(2,3,2)) = 13 \]
2. Step: 
\[4 \leq \text{delay} \leq 7 \rightarrow \delta = 5\]

Mark vertices \( u \) as “good“ if:
\[\text{distFromHome}[u] + \text{distToJanet}[u] \leq A + \delta \text{ and } \text{distToJanet}[u] \leq \delta\]

- \( u_1: \) \[0 + 7 \leq 10 + 5 \text{ and } 7 \leq 5\]
- \( u_2: \) \[11 + 2 \leq 10 + 5 \text{ and } 7 \leq 5\]
- \( u_3: \) \[7 + 0 \leq 10 + 5 \text{ and } 7 \leq 5\]
2. Step:
$4 \leq \text{delay} \leq 7 \rightarrow \delta = 5$

Propagatge: if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}[v] \leq \delta$, then mark $v$ and edge $l$ as good too

$u_2, e(2,1,1): \quad 1 + 7 \leq 5$
$u_2, e(2,3,2): \quad 2 + 0 \leq 5$
$u_2, e(2,3,5): \quad 5 + 0 \leq 5$
$u_3, e(3,2,4): \quad 4 + 2 \leq 5$
2. Step:
\[4 \leq \text{delay} \leq 7 \rightarrow \delta = 5\]

If subgraph of good edges has cycle \(\rightarrow\) delay \(\delta\) is possible
\(\text{hasCycle()} = \text{false}\)
2. Step:
4 ≤ delay ≤ 7 → δ = 5

Compute longest time Richard can stay in the subgraph
If this is ≥ B then delay δ is possible
latest[u] = A + delay − distToJanet[u]

latest[2] = 10 + 5 − 2 = 13
latest[3] = 10 + 5 − 0 = 15

Start with good nodes with indeg[u] == 0
Check if: latest[u] + distToJanet[u] ≥ B

Problem: Well Spoken

### Approach - Example

3. Step:
6 ≤ delay ≤ 7 → δ = 6

Mark vertices u as “good“ if:
\[ \text{distFromHome}[u] + \text{distToJanet}[u] ≤ A + δ \text{ and } \text{distToJanet}[u] ≤ δ \]

- **u₁:** 0 + 7 ≤ 10 + 6 and 7 ≤ 6
- **u₂:** 11 + 2 ≤ 10 + 6 and 7 ≤ 6
- **u₃:** 7 + 0 ≤ 10 + 6 and 7 ≤ 6
3. Step:
6 ≤ delay ≤ 7 → δ = 6

Propagate: if u is good and \( u \rightarrow v \) with \( l + \text{distToJanet}[v] ≤ δ \),
then mark v and edge l as good too

\[
\begin{align*}
    u_2, e(2,1,1): & \quad 1 + 7 ≤ 6 \\
    u_2, e(2,3,2): & \quad 2 + 0 ≤ 6 \\
    u_2, e(2,3,5): & \quad 5 + 0 ≤ 6 \\
    u_3, e(3,2,4): & \quad 4 + 2 ≤ 6
\end{align*}
\]
3. Step:
6 ≤ delay ≤ 7 → δ = 6

If subgraph of good edges has cycle → delay δ is possible
hasCycle() = true
4. Step:
6 ≤ delay ≤ 6 → left == right
→ Output := 6
Problem L: Well Spoken

**Runtime**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approach</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>WellSpoken()</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Dijkstra**: $O(E + V \log V)$
- **BinarySearch**: $O(\log w) \times O(?)$

with $w = distToJanet(s)$
**checkDelay(δ)**

1. Mark vertices $u$ as “good“ if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too

3. If subgraph of good edges has cycle $\rightarrow$ delay $\delta$ is possible

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible
checkDelay(δ)

1. Mark vertices $u$ as “good“ if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too

3. If subgraph of good edges has cycle $\rightarrow$ delay $\delta$ is possible

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible
Problem L: Well Spoken

CheckDelay(δ)

1. Mark vertices u as “good“ if distFromHome(u) + distToJanet(u) ≤ A + δ and distToJanet(u) ≤ δ  \(O(V)\)

2. Propagate: if u is good and \(u \rightarrow v\) with \(l + distToJanet(v) ≤ δ\), then mark v and edge l as good too  \(O(E)\)

3. If subgraph of good edges has cycle \(\rightarrow\) delay δ is possible

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is \(≥ B\) then delay δ is possible

Runtime

<table>
<thead>
<tr>
<th>Problem</th>
<th>Approach</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
checkDelay($\delta$)

1. Mark vertices $u$ as "good" if $distFromHome(u) + distToJanet(u) \leq A + \delta$ and $distToJanet(u) \leq \delta$  \[ O(V) \]

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + distToJanet(v) \leq \delta$, then mark $v$ and edge $l$ as good too  \[ O(E) \]

3. If subgraph of good edges has cycle $\rightarrow$ delay $\delta$ is possible  \[ O(V + E) \]

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible
**Problem L: Well Spoken**

**checkDelay(δ)**

1. Mark vertices $u$ as “good“ if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$.  

   $O(V)$

2. Propagate: if $u$ is good and $u \xrightarrow{l} v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too.  

   $O(E)$

3. If subgraph of good edges has cycle $\xrightarrow{}$ delay $\delta$ is possible  

   $O(V + E)$

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible  

   $O(\?)$
Runtime – `longestPath()`

### Pseudocode:

```java
longestPath(δ)

    while(!goodNodes.isEmpty())
        u = goodNodes.remove(0)
        if latest[u] + distToJanet[u] ≥ B
            return true
        for Edge e in goodEdges[u]
            indeg[e.dest] -= 1
            if indeg[e.dest] == 0
                goodNodes.add(e.dest)
            latest[e.dest] = max(latest[e.dest], latest[u] + e.weight)

    return false
```

Problem L: Well Spoken

Hendrik Meininger, Johannes Schleicher
### Runtime – longestPath()

#### Pseudocode:

**longestPath(δ)**

```plaintext
while(!goodNodes.isEmpty())
    u = goodNodes.remove(0)
    if latest[u] + distToJanet[u] ≥ B
        return true

for Edge e in goodEdges[u]
    indeg[e.dest] += 1
    if indeg[e.dest] == 0
        goodNodes.add(e.dest)
    latest[e.dest] = max(latest[e.dest], latest[u] + e.weight)
```

Return false

---

**Runtime**

- `longestPath()`

**Problem Approach**

**Problem L: Well Spoken**

Hendrik Meininger, Johannes Schleicher
Problem L: Well Spoken

Hendrik Meininger, Johannes Schleicher

Runtime – longestPath()

Pseudocode:

`longestPath(δ)`

```java
while(!goodNodes.isEmpty())
    u = goodNodes.remove(0)
    if latest[u] + distToJanet[u] ≥ B
        return true
    for Edge e in goodEdges[u]
        indeg[e.dest] -= 1
        if indeg[e.dest] == 0
            goodNodes.add(e.dest)
        latest[e.dest] = max(latest[e.dest], latest[u] + e.weight)
return false
```

Results:
- Runs in time complexity $O(V + E)$
### CheckDelay(\(\delta\))

1. Mark vertices \(u\) as “good” if \(\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta\) and \(\text{distToJanet}(u) \leq \delta\)

   \(O(V)\)

2. Propagate: if \(u\) is good and \(u \rightarrow v\) with \(l + \text{distToJanet}(v) \leq \delta\), then mark \(v\) and edge \(l\) as good too

   \(O(E)\)

3. If subgraph of good edges has cycle \(\rightarrow\) delay \(\delta\) is possible

   \(O(V + E)\)

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is \(\geq B\) then delay \(\delta\) is possible

   \(O(V + E)\)
### checkDelay(δ)

1. Mark vertices $u$ as "good" if $\text{distFromHome}(u) + \text{distToJanet}(u) \leq A + \delta$ and $\text{distToJanet}(u) \leq \delta$
   \[ O(V) \]

2. Propagate: if $u$ is good and $u \rightarrow v$ with $l + \text{distToJanet}(v) \leq \delta$, then mark $v$ and edge $l$ as good too
   \[ O(E) \]

3. If subgraph of good edges has cycle $\rightarrow$ delay $\delta$ is possible
   \[ O(V + E) \]

4. Otherwise the subgraph of good nodes and edges is acyclic. Compute longest time Richard can stay in the subgraph. If this is $\geq B$ then delay $\delta$ is possible
   \[ O(V + E) \]

\[ \rightarrow O(V + E) \]
Runtime:

WellSpoken()

- **Dijkstra**
  - $O(E + V \log V)$

- **BinarySearch**
  - $O(V + E)$
  - $O(\log w) \times O(V + E)$
    - with $w = \text{distToJanet}(s)$

*Runtime: $O((V + E) \times \log(w))$*