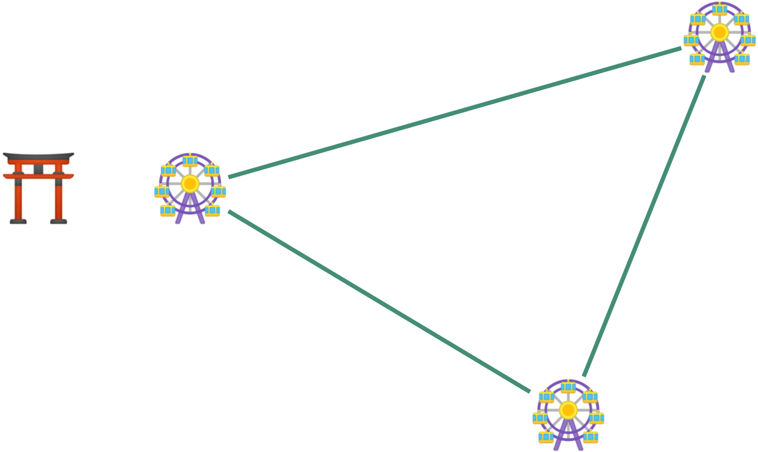
Problem Q: A Stingy Park Visit

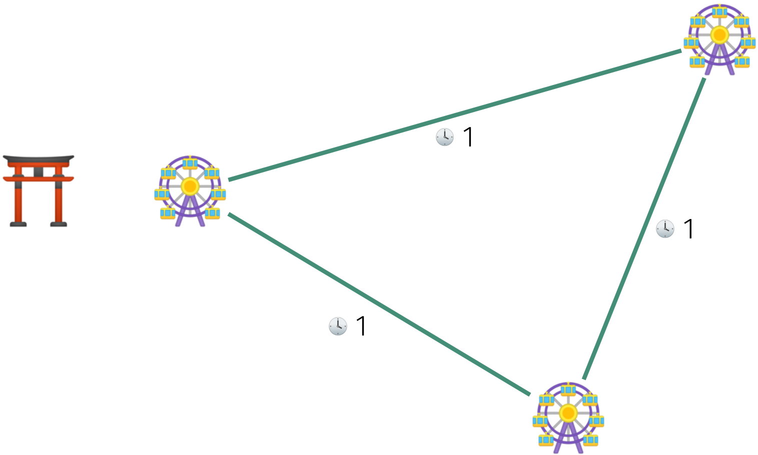
Seminar Algorithms for Programming Contests

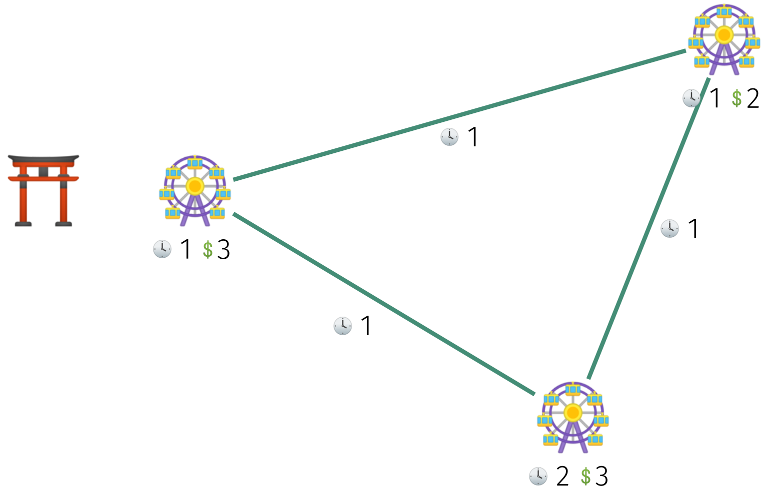
Tim Hegemann Michael Kreuzer

2020-07-22

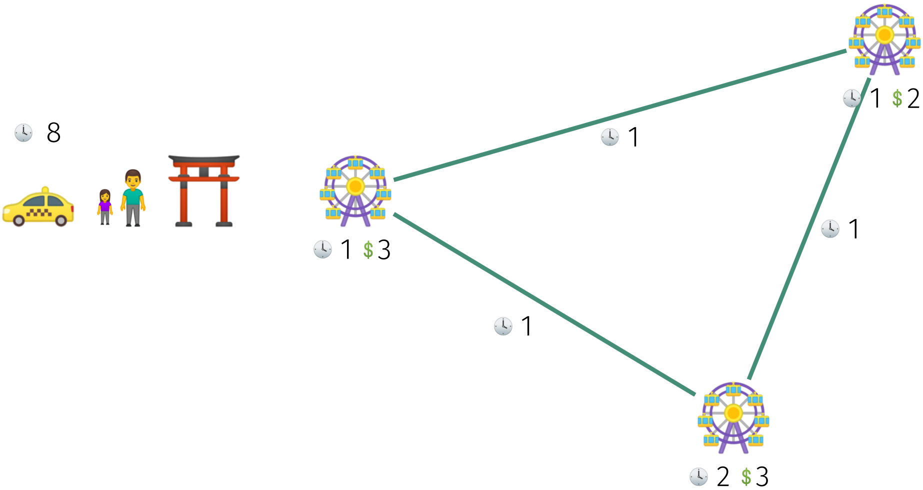

Tim
Michael

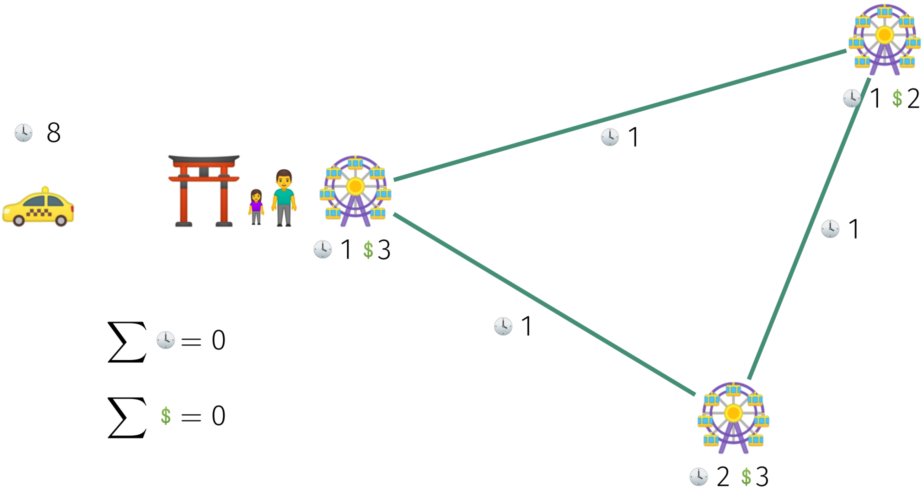


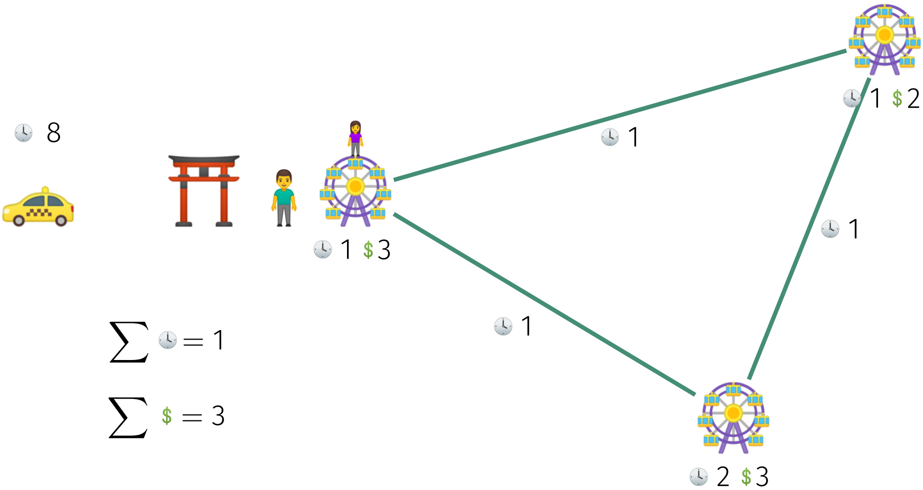

1
1
1

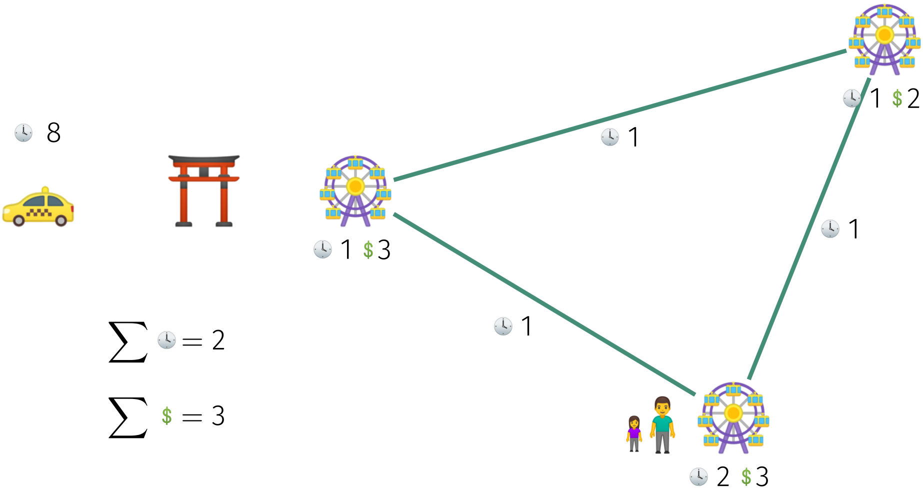


1 $ 3
1 $ 2
2 $ 3
1
1
1

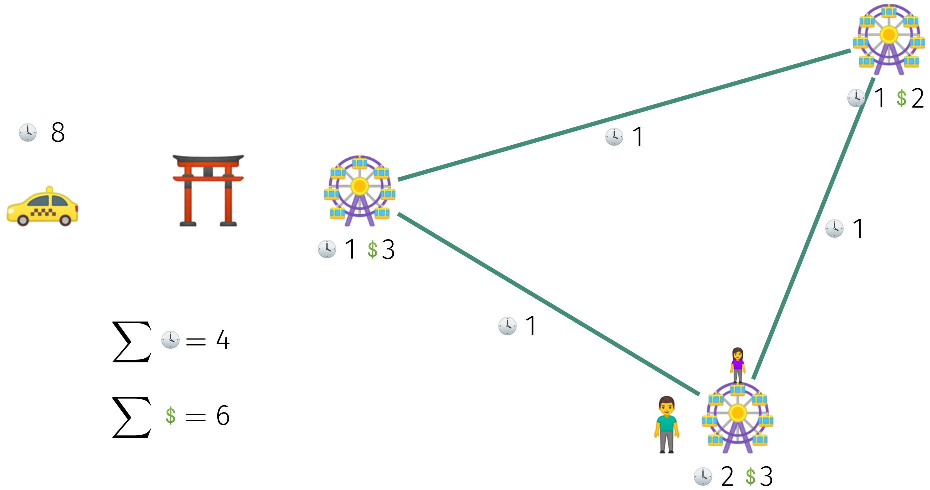

8
1 $3
1 $2
2 $3
1
1
1

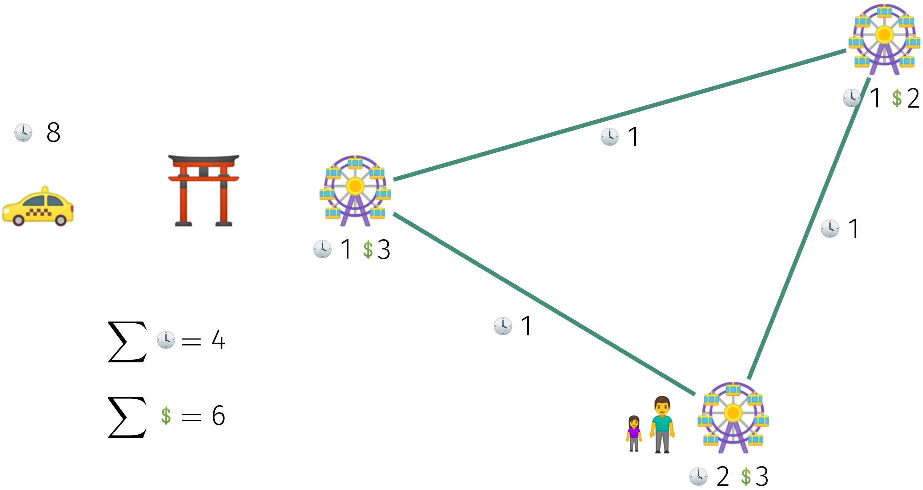

🕓 8
🕓 1 $ 3
🕓 1
🕓 1 $ 2
🕓 1
🕓 1
🕓 2 $ 3
∑ 🕓 = 0
∑ $ = 0


8
1 $3
1
1 $2
1
1
2 $3
∑ = 1
∑ $ = 3

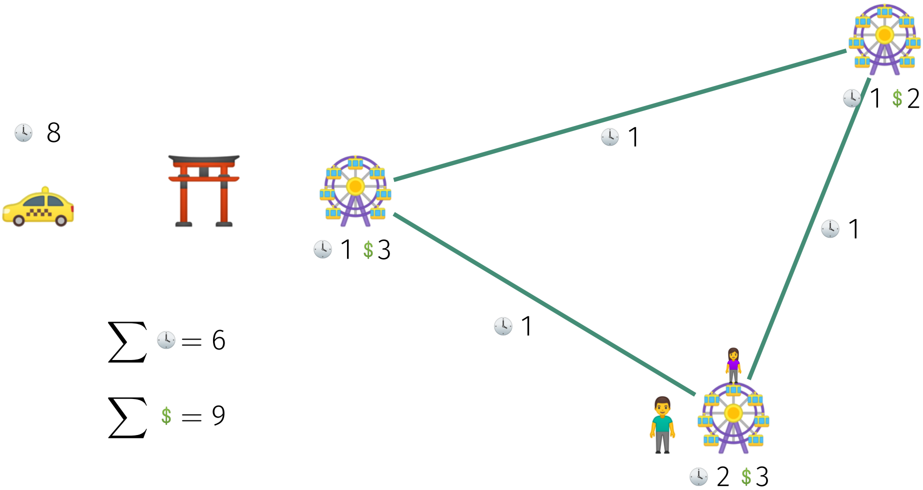


8
1 $3
1 $2
1
1
1
2 $3
∑ = 2
∑ $ = 3


8
1 $3
1
1 $2
1
1
2 $3
∑ = 4
∑ $ = 6

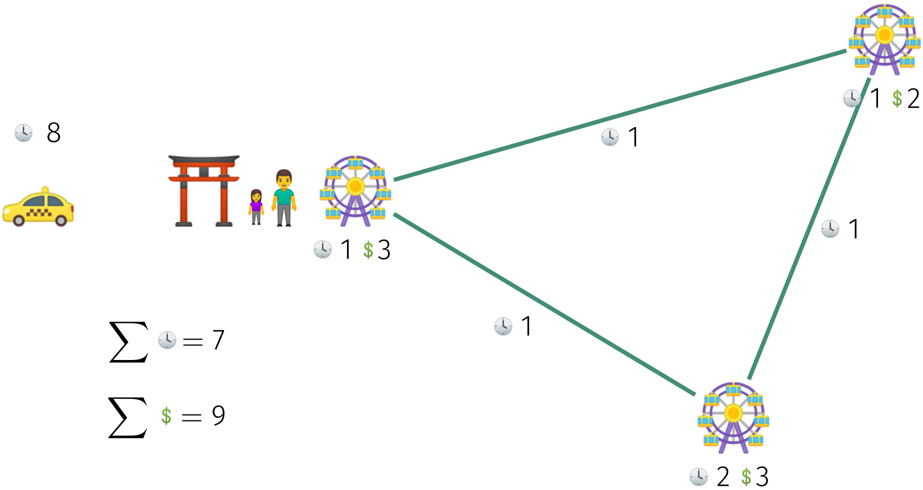


🕓 8
🕓 1 $ 3
🕓 1
🕓 1 $ 2
🕓 1
🕓 1
🕓 2 $ 3
∑ 🕓 = 4
∑ $ = 6

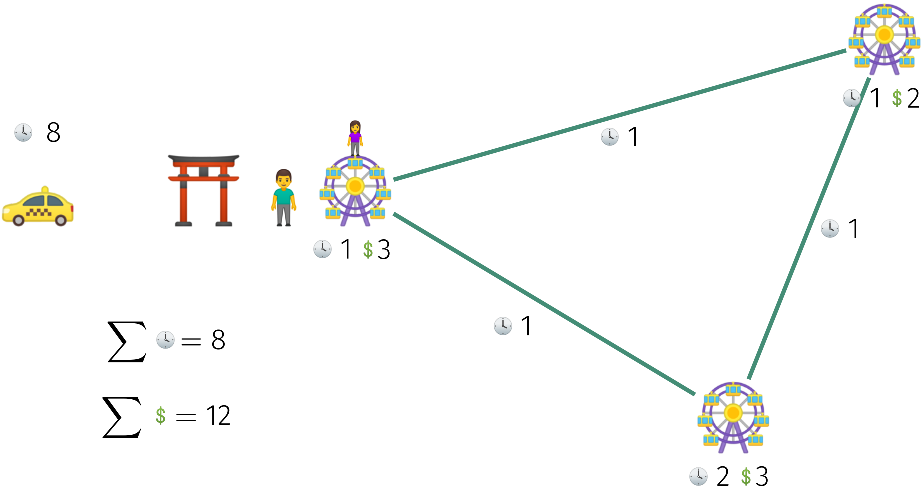


$$\sum 🕓 = 6$$

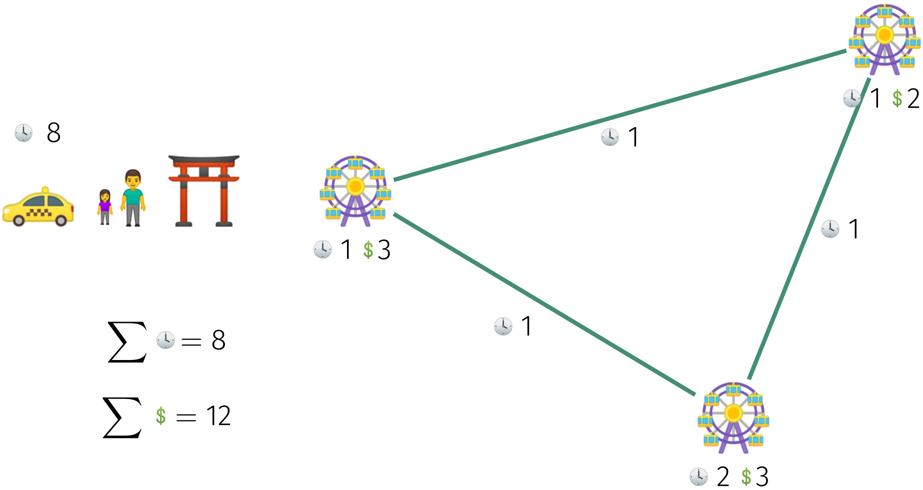
$$\sum \$ = 9$$



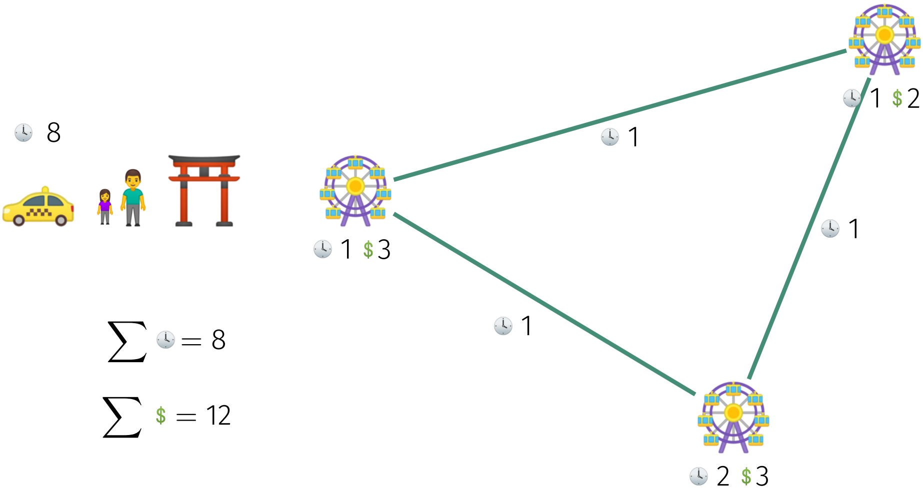
8
1 $3
1 $2
1 $2
1
1
1
2 $3
∑ = 7
∑ $ = 9



8
1 $3
1
1 $2
1
1
2 $3
$\sum = 8$
$\sum \$ = 12$



🕓 8
🕓 1 $ 3
🕓 1 $ 2
🕓 2 $ 3
🕓 1
🕓 1
🕓 1
∑ 🕓 = 8
∑ $ = 12



$$\sum 🕓 = 8$$

$$\sum \$ = 12$$

all emoji: Google Noto Fonts · SIL OFL

A Stingy Park Visit

It is another wonderful sunny day in July – and you decided to spend your day together with your little daughter Joy. Since she really likes the fairy-park in the next town, you decided to go there for the day. Your wife (unfortunately she has to work) agreed to drive you to the park and pick you up again. Alas, she is very picky about being on time, so she told you exactly when she will be at the park's front entrance to pick you up and you have to be there at exactly that time. You clearly also don't want to wait outside, since this would make your little daughter sad – she could have spent more time in the park!

A Stingy Park Visit (cont.)

Now you have to plan your stay at the park. You know when you will arrive and when you will have to depart. The park consists of several rides, interconnected by small pavements. The entry into the park is free, but you have to pay for every use of every ride in the park. Since it is Joy's favorite park, you already know how long using each ride takes and how much each ride costs. When walking through the park, you obviously must not skip a ride when walking along it (even if Joy has already used it), or else Joy would be very sad. Since Joy likes the park very much, she will gladly use rides more than once. Walking between two rides takes a given amount of time.
Since you are a provident parent you want to spend as little as possible when being at the park. Can you compute how much is absolutely necessary?

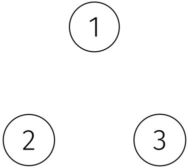
```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

x time to spend in the park

n number of rides

m number of pavements

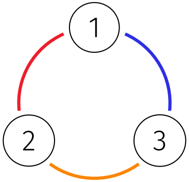
t time needed to pass each pavement



```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

x	time to spend in the park
n	number of rides
m	number of pavements
t	time needed to pass each pavement

m lines each defining a pavement between two rides.

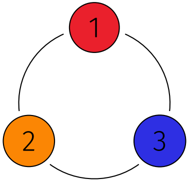


```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

x	time to spend in the park
n	number of rides
m	number of pavements
t	time needed to pass each pavement

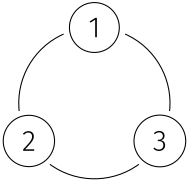
m lines each defining a pavement between two rides.

n lines each defining time and cost of a ride.



```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

Output: minimum cost for a resulting path if possible or »`It is a trap.`« else.

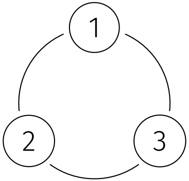


9

```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

Output: minimum cost for a resulting path if possible or »`It is a trap.`« else.

$$1 \leq x, n, m, t \leq 1\,000$$




```
9
```

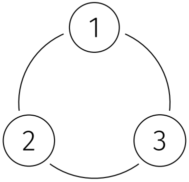
```
6
3 3 1
1 2
2 3
3 1
1 3
1 2
2 3
```

```
9
```

Output: minimum cost for a resulting path if possible or »`It is a trap.`« else.

$$1 \leq x, n, m, t \leq 1\,000$$

$$1 \leq t_\nu, p_\nu \leq 10^6 \qquad \forall \nu : 1 \leq \nu \leq n$$



Recap:
There are a number of rides   that we may or may not take.
Each one has a time   constraint on them and we only have given time.
Each has a price   that we seek to minimize.

Does that remind you of anything?

Recap:

There are a number of rides **items** that we may or may not take.

Each one has a time **weight** constraint on them and we only have given time. **can only carry given weight**

Each has a price **value** that we seek to minimize. **maximize**

Does that remind you of anything?

$$f(\xi,\nu) = \text{the minimum cost of any tour from start to ride } \nu \text{ that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \text{the minimum cost of any tour from start to ride } \nu \text{ that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \begin{cases} \\ \\ \end{cases}$$

$$f(\xi, \nu) = \text{the minimum cost of any tour from start to ride } \nu$$
$$\text{that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \begin{cases} p_{start} & \text{if } \xi = t_{start} \land \nu = \text{start} \\ & \end{cases}$$

$$f(\xi, \nu) = \text{the minimum cost of any tour from start to ride } \nu$$
$$\text{that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \begin{cases} p_{start} & \text{if } \xi = t_{start} \wedge \nu = \text{start} \\ \infty & \text{if } \xi \leq t_{start} \\ & \end{cases}$$

$$f(\xi, \nu) = \text{the minimum cost of any tour from start to ride } \nu \text{ that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \begin{cases} p_{start} & \text{if } \xi = t_{start} \land \nu = \text{start} \\ \infty & \text{if } \xi \leq t_{start} \\ \min \{p_\nu + f(\xi - t_\nu, \nu)\} \cup \{p_\nu + f(\xi - t - t_\nu, \iota) \mid \iota \in Adj(\nu)\} \end{cases}$$

$$
\begin{aligned}
f(\xi, \nu) = {} & \text{the minimum cost of any tour from start to ride } \nu \\
& \text{that takes exactly } \xi \text{ time.}
\end{aligned}
$$

$$
f(\xi, \nu) = \begin{cases} p_{start} & \text{if } \xi = t_{start} \wedge \nu = \text{start} \\ \infty & \text{if } \xi \leq t_{start} \\ \min \{p_\nu + f(\xi - t_\nu, \nu)\} \cup \{p_\nu + f(\xi - t - t_\nu, \iota) \mid \iota \in Adj(\nu)\} \end{cases}
$$

ride again

$$f(\xi, \nu) = \text{the minimum cost of any tour from start to ride } \nu$$
$$\text{that takes exactly } \xi \text{ time.}$$

$$f(\xi, \nu) = \begin{cases} p_{start} & \text{if } \xi = t_{start} \wedge \nu = \text{start} \\ \infty & \text{if } \xi \leq t_{start} \\ \min \{p_\nu + f(\xi - t_\nu, \nu)\} \cup \{p_\nu + f(\xi - t - t_\nu, \iota) \mid \iota \in Adj(\nu)\} \end{cases}$$

ride again

take a path and ride once

var *price* = array[x][n]
price.fill(∞)
price[t_{start}][1] = p_{start}

var *price* = array[x][n]
price.fill(∞)
price[t_{start}][1] = p_{start}
for $\xi \in [1..x]$ **do**

var *price* = array[x][n]
price.fill(∞)
price[t_{start}][1] = p_{start}
for $\xi \in$ *[1..x]* **do**
    for *every ride* ν **do**
        price[ξ][ν] = price[$\xi - t_\nu$][ν] $+ p_\nu$ } ride again

```
var price = array[x][n]
price.fill(∞)
price[t_start][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
```

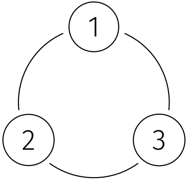
ride again

take a path
and ride once

```
var price = array[x][n]
price.fill(∞)
price[t_start][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

ride again

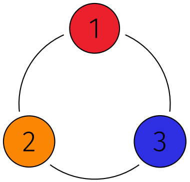
take a path
and ride once



$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$\xi \setminus \nu$	1	2	3
1			
2			
3			
4			
5			
6			

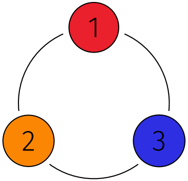


init

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2			
3			
4			
5			
6			

$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

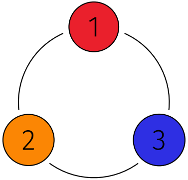


$p_1 + f(\xi - t_1, 1)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6,		
3			
4			
5			
6			

$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3



$t = 1, \; x = 6$

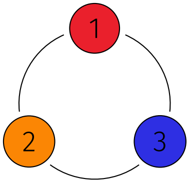
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

$p_1 + f(\xi - t - t_1, 2)$

$\lightning$ out of bounds

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6, ∞,		
3			
4			
5			
6			



$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

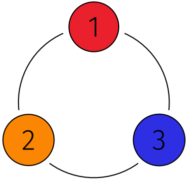
$p_1 + f(\xi - t - t_1, 2)$

↯ out of bounds

$p_1 + f(\xi - t - t_1, 3)$

↯ out of bounds

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6, ∞, ∞		
3			
4			
5			
6			

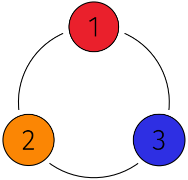


$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞,	
3			
4			
5			
6			



$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

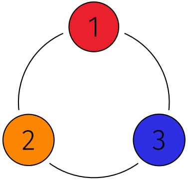
$p_2 + f(\xi - t - t_2, 1)$

↯ out of bounds

$p_2 + f(\xi - t - t_2, 3)$

↯ out of bounds

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞, ∞, ∞	
3			
4			
5			
6			



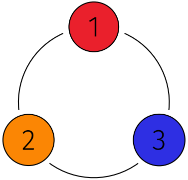
$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_3 + f(\xi - t_3, 3)$

$\lightning$ out of bounds

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3			
4			
5			
6			

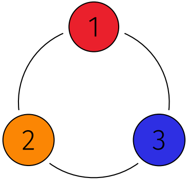


$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9,		
4			
5			
6			



$t = 1, \; x = 6$

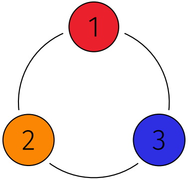
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

$p_1 + f(\xi - t - t_1, 2)$

$p_1 + f(\xi - t - t_1, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9, ∞, ∞		
4			
5			
6			

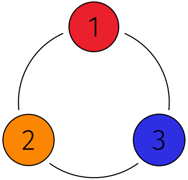


$t = 1,\; x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	∞,	
4			
5			
6			



$t = 1,\ x = 6$

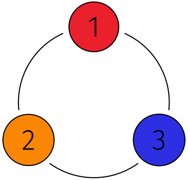
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

$p_2 + f(\xi - t - t_2, 1)$

$p_2 + f(\xi - t - t_2, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	∞, 5, ∞	
4			
5			
6			



$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_3 + f(\xi - t_3, 3)$

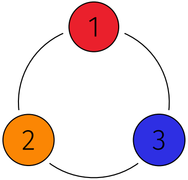
$p_3 + f(\xi - t - t_3, 1)$

↯ out of bounds

$p_3 + f(\xi - t - t_3, 2)$

↯ out of bounds

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞, ∞, ∞
4			
5			
6			



$t = 1,\ x = 6$

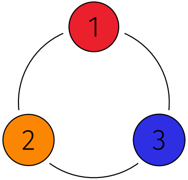
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$$p_1 + f(\xi - t_1, 1)$$

$$p_1 + f(\xi - t - t_1, 2)$$

$$p_1 + f(\xi - t - t_1, 3)$$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12, ∞, ∞		
5			
6			



$t = 1,\ x = 6$

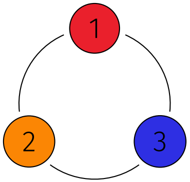
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

$p_2 + f(\xi - t - t_2, 1)$

$p_2 + f(\xi - t - t_2, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7, 8, ∞	
5			
6			



$t = 1,\ x = 6$

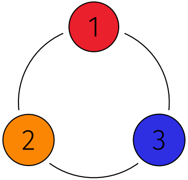
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_3 + f(\xi - t_3, 3)$

$p_3 + f(\xi - t - t_3, 1)$

$p_3 + f(\xi - t - t_3, 2)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	∞, 6, ∞
5			
6			



$t = 1,\; x = 6$

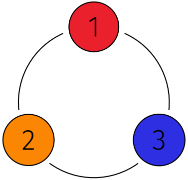
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

$p_1 + f(\xi - t - t_1, 2)$

$p_1 + f(\xi - t - t_1, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	6
5	15, 8, ∞		
6			



$t = 1,\ x = 6$

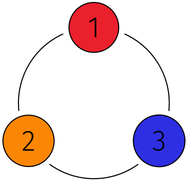
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_2 + f(\xi - t_2, 2)$

$p_2 + f(\xi - t - t_2, 1)$

$p_2 + f(\xi - t - t_2, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	6
5	8	9, 11, ∞	
6			



$t = 1,\ x = 6$

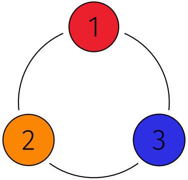
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_3 + f(\xi - t_3, 3)$

$p_3 + f(\xi - t - t_3, 1)$

$p_3 + f(\xi - t - t_3, 2)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	6
5	8	9	∞, 9, ∞
6			



$t = 1,\; x = 6$

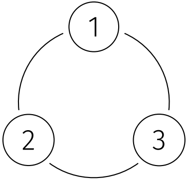
ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

$p_1 + f(\xi - t_1, 1)$

$p_1 + f(\xi - t - t_1, 2)$

$p_1 + f(\xi - t - t_1, 3)$

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	6
5	8	9	9
6	11, 10, 9		



$t = 1,\ x = 6$

ν	1	2	3
t_ν	1	1	2
p_ν	3	2	3

done!

$\xi \setminus \nu$	1	2	3
1	3	∞	∞
2	6	∞	∞
3	9	5	∞
4	12	7	6
5	8	9	9
6	9		

```
var price = array[x][n]
price.fill(∞)
price[t][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

```
var price = array[x][n]
price.fill(∞)
price[t][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

$\mathcal{O}(\deg(\nu))$

```
var price = array[x][n]
price.fill(∞)
price[t][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(                              𝒪(n + m)
                price[ξ][ν],                      𝒪(deg(ν))
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

```
var price = array[x][n]
price.fill(∞)
price[t][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do                      } O(deg(ν))   } O(n + m)   } O(x * (n + m))
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

$\mathcal{O}(\deg(\nu))$

$\mathcal{O}(n+m)$

$\mathcal{O}(x*(n+m))$

```
var price = array[x][n]        O(x * n)
price.fill(∞)
price[t][1] = p_start
for ξ ∈ [1..x] do
    for every ride ν do
        price[ξ][ν] = price[ξ − t_ν][ν] + p_ν
        for every neighbour ι of ν do
            price[ξ][ν] = min(
                price[ξ][ν],
                price[ξ − t_ν − t][ι] + p_ν
            )
return price[x][1]
```

$\mathcal{O}(\deg(\nu))$

$\mathcal{O}(n + m)$

$\mathcal{O}(x * (n + m))$

var *price* = array[x][n] $\mathcal{O}(x * n)$
price.fill(∞)
price[t][1] = p_{start}
for $\xi \in [1..x]$ **do**
    for *every ride* ν **do**
        price[ξ][ν] = price[$\xi - t_\nu$][ν] $+ p_\nu$
        for *every neighbour* ι *of* ν **do** $\mathcal{O}(m)$
            price[ξ][ν] = min(
                price[ξ][ν],
                price[$\xi - t_\nu - t$][ι] $+ p_\nu$
)
return price[x][1]

$\mathcal{O}(\deg(\nu))$

$\mathcal{O}(n + m)$

$\mathcal{O}(x * (n + m))$

var *price* = array[x][n] $\mathcal{O}(x * n)$

price.fill(∞)

price[t][1] = p_{start}

for $\xi \in [1..x]$ **do**

    for *every ride* ν **do**

        price[ξ][ν] = price[$\xi - t_\nu$][ν] $+ p_\nu$

        for *every neighbour* ι *of* ν **do** $\mathcal{O}(m)$

            price[ξ][ν] = min(

                price[ξ][ν],

                price[$\xi - t_\nu - t$][ι] $+ p_\nu$

) $\mathcal{O}(\deg(\nu))$ $\mathcal{O}(n + m)$

$\mathcal{O}(x * (n + m))$

$\mathcal{O}(m + (x * n))$

return price[x][1]

Benchmarks

For our algorithm we proved

- runtime in $\mathcal{O}((n + m) * x)$
- memory demand in $\mathcal{O}(m + n * x)$

Benchmarks

For our algorithm we proved

- runtime in $\mathcal{O}((n + m) * x)$
- memory demand in $\mathcal{O}(m + n * x)$

m and n are not independent.

$$m \leq \frac{n * (n - 1)}{2}$$

Benchmarks

For our algorithm we proved

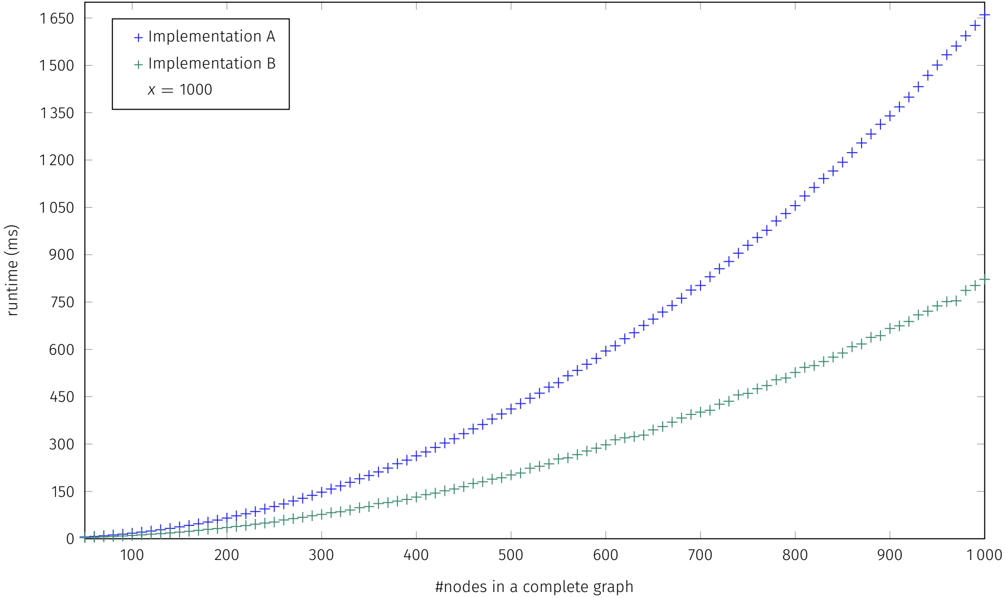
- runtime in $\mathcal{O}((n+m)*x)$
- memory demand in $\mathcal{O}(m+n*x)$

m and n are not independent.

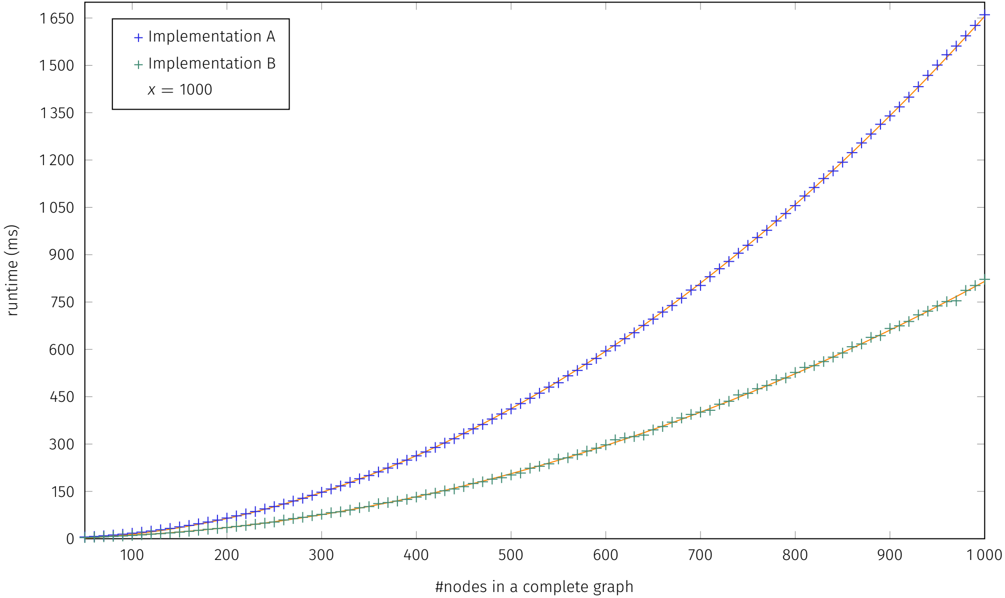
$$m = \frac{n*(n-1)}{2}$$

Runtime depends on $deg(\nu)$. Worst case: complete graphs.
For our implementations we expect

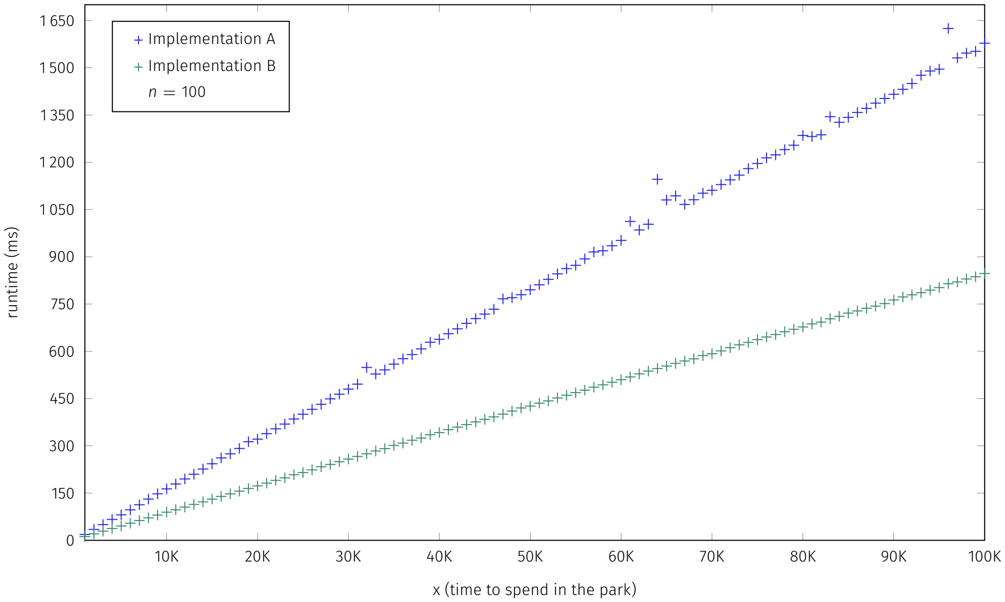
- runtime and memory demand in $\mathcal{O}(n^2)$ for fixed x
- runtime and memory demand in $\mathcal{O}(x)$ for fixed n

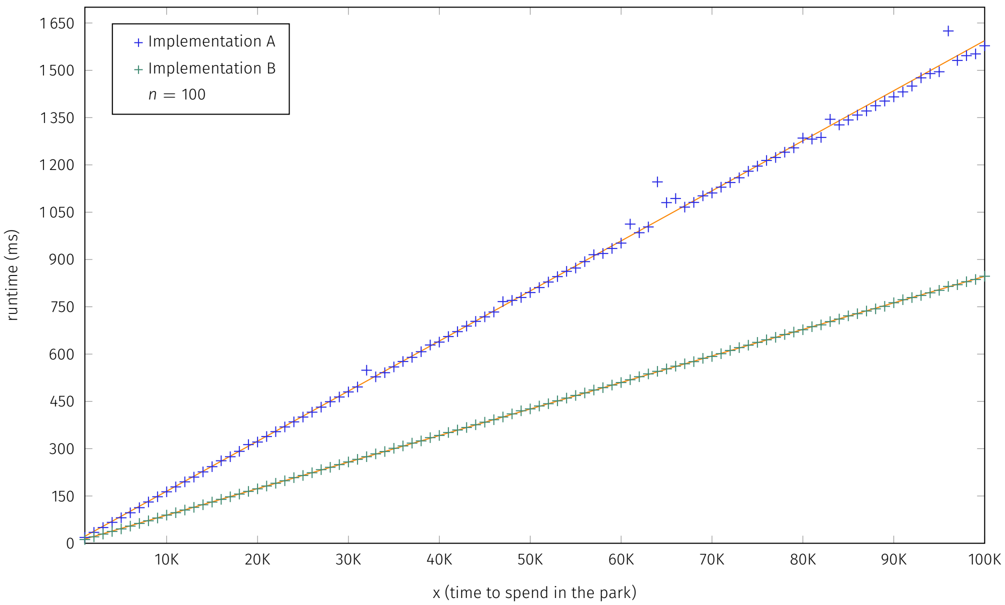


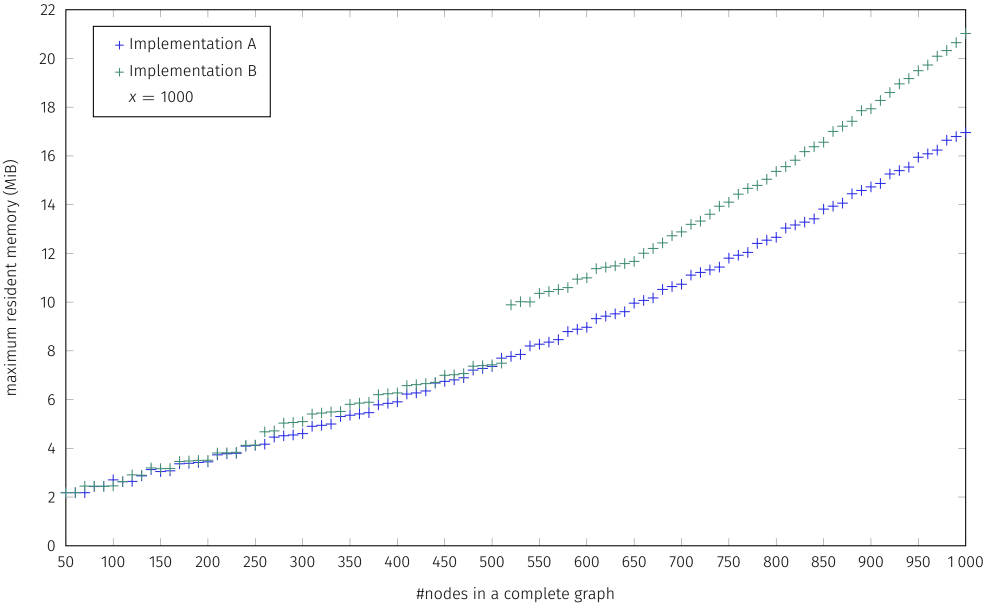
Implementation A
Implementation B
x = 1000
runtime (ms)
#nodes in a complete graph
0
150
300
450
600
750
900
1 050
1 200
1 350
1 500
1 650
100
200
300
400
500
600
700
800
900
1 000



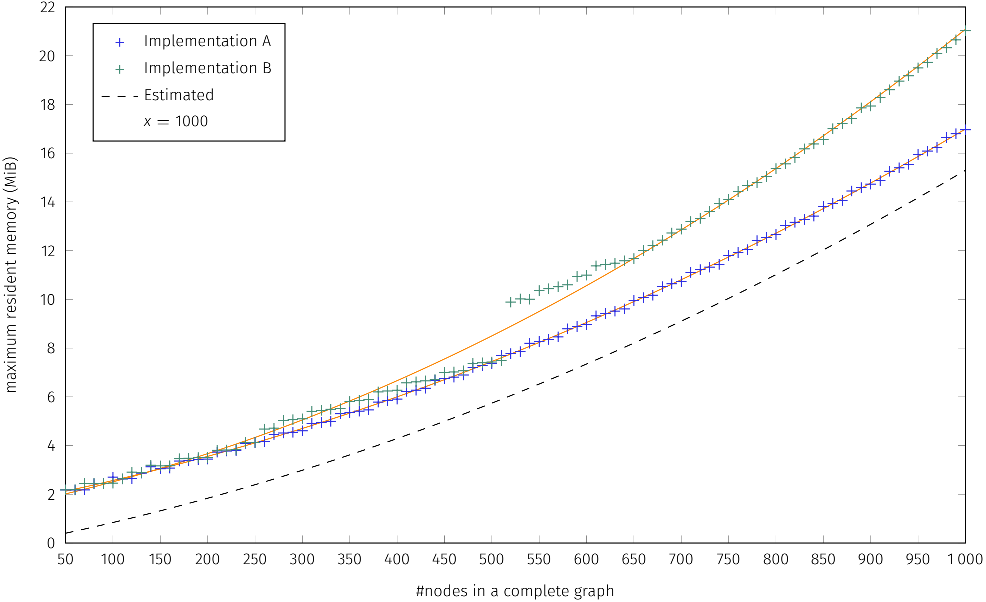
Implementation A
Implementation B
$x = 1000$
runtime (ms)
0
150
300
450
600
750
900
1 050
1 200
1 350
1 500
1 650
100
200
300
400
500
600
700
800
900
1 000
#nodes in a complete graph


Implementation A
Implementation B
n = 100
runtime (ms)
x (time to spend in the park)
0
150
300
450
600
750
900
1 050
1 200
1 350
1 500
1 650
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K

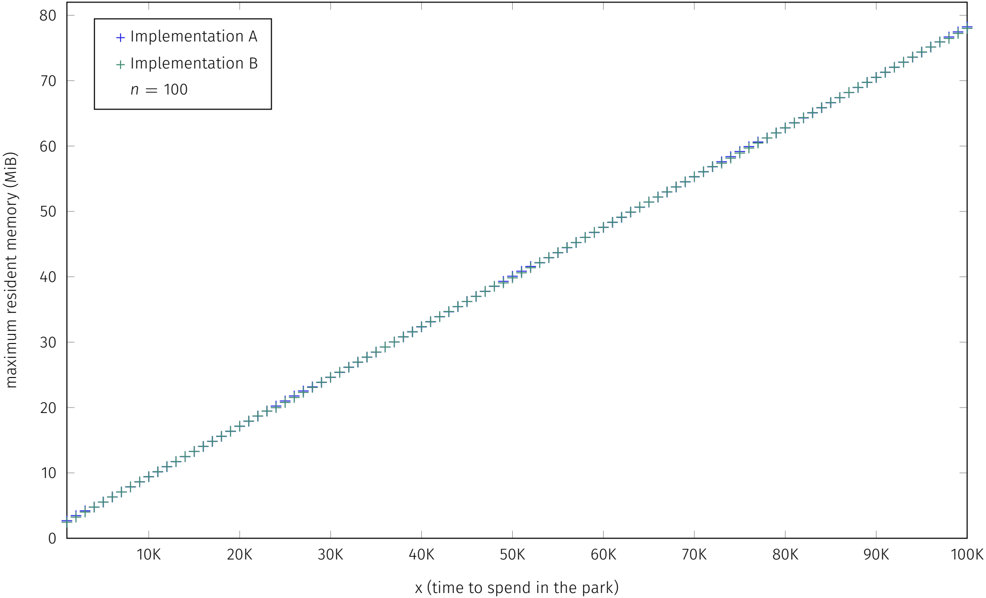

Implementation A
Implementation B
$n = 100$
runtime (ms)
x (time to spend in the park)
0
150
300
450
600
750
900
1 050
1 200
1 350
1 500
1 650
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K



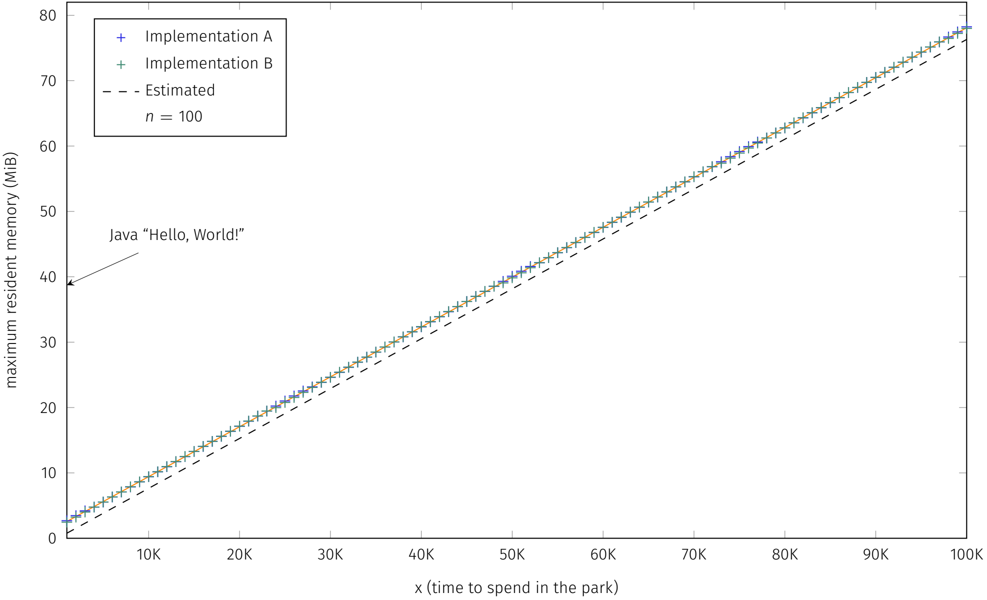
Implementation A
Implementation B
x = 1000
maximum resident memory (MiB)
0
2
4
6
8
10
12
14
16
18
20
22
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1 000
#nodes in a complete graph



Implementation A
Implementation B
Estimated
$x = 1000$
maximum resident memory (MiB)
#nodes in a complete graph
0
2
4
6
8
10
12
14
16
18
20
22
50
100
150
200
250
300
350
400
450
500
550
600
650
700
750
800
850
900
950
1 000



Implementation A
Implementation B
$n = 100$
maximum resident memory (MiB)
x (time to spend in the park)
0
10
20
30
40
50
60
70
80
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K



Implementation A
Implementation B
Estimated
$n = 100$
Java “Hello, World!”
maximum resident memory (MiB)
x (time to spend in the park)
0
10
20
30
40
50
60
70
80
10K
20K
30K
40K
50K
60K
70K
80K
90K
100K

How We Measure Performance

Variation

- At least 100 repetitions per parameter
- Coefficient of variation $< 2.79\%$ for 95% of our measurements

Tools

- Runtime: hyperfine
- Memory: GNU Time (the name is a bit misleading)

Platform

- Fedora 32 (Kernel 5.7.8)
- AMD Ryzen 9 3900X
- 64 GiB RAM

Memory Estimation

- Nodes: $(5 + deg(\nu)) * 8byte$ per node $\Rightarrow (5n + 2m) * 8byte$ total
- Table: $(x * n) * 8byte$

Our implementation needs at least $((x + 5) * n + 2m) * 8byte$ of memory.

Complete graph with n nodes, $x = 1000$:	$(8n^2 + 8040n) * 1024^{-2}$ MiB
K100:	$(800x + 10500) * 1024^{-2}$ MiB

Regression Analysis

Complete graphs with n nodes, $x = 1000$ (runtime in ms)

- Implementation A: $1.65762 * 10^{-3} * n^2 - 2.19840$
- Implementation B: $8.12437 * 10^{-4} * n^2 + 3.04082$

K100, x (runtime in ms)

- Implementation A: $0.0158775 * x + 6.35848$
- Implementation B: $0.00841535 * x + 5.05326$

Complete graphs with n nodes, $x = 1000$ (memory in MiB)

- Implementation A: $7.63536 * 10^{-6} * n^2 + 0.00762691 * n + 1.72713$
- Implementation B: $1.13240 * 10^{-5} * n^2 + 0.00817827 * n + 1.58061$

K100, x (memory in MiB)

- Implementation A: $7.62959 * 10^{-4} * x + 1.80158$
- Implementation B: very similar to A

Viel Spaß!