Segundo Trabalho de Grupo (T2)

Agência de Viagens

Trabalho realizado por:

Alexandre Correia - 202007042
João Araújo - 202007855
Tiago Branquinho – 202005567

Grupo : G15

Descrição do problema

Uma empresa de transportes desenvolveu uma plataforma eletrónica para a realização de viagens turísticas. A empresa dispõe de veículos em vários locais. Cada um fará um único trajeto de uma origem para um destino, transportando uma certa quantidade de pessoas num certo tempo (dado em horas) e com um custo de transporte (bilhete) por pessoa.

Pretende-se um sistema capaz de apoiar a gestão de pedidos para transporte de grupos de pessoas de um local de origem para um local de destino.

Cenário 1: Grupos não se separam

Formalização 1.1 :

Pretende-se implementar um algoritmo que indique ao grupo um possível encaminhamento de o local de origem para o local de destino.

Função objetivo: Max C = Min(C_a) ∧ a ∈ {o, ..., d}, sendo C a capacidade e a uma aresta do grafo.

A capacidade do percurso (Max C), é, neste caso, o minimo das capacidades dos ramos que constituem o percurso (com maior mínimo).

Restrições: C = [0, maior peso do conjunto de arestas(MP)] = {C ∈ IR: 0 < C < MP}

Input :

- Nó(inicial), Ni ∈ [1, tamanho_grafo] (ni ∈ IN)
- Nó(final), Nf ∈ [1, tamanho_grafo] (nf ∈ IN)

Output:

- Capacidade máxima
- Caminho entre o nó inicial e o nó final

Algoritmos utilizados e complexidades:

Algoritmo	Temporal	Espacial
1.1 - Caminhos de capacidade máxima (adaptação do Algoritmo de Dijkstra) através de lista de adjacências usando uma fila de prioridade suportada por um heap máximo.	O((|V| + |E|) log2 |V|)	O(1)

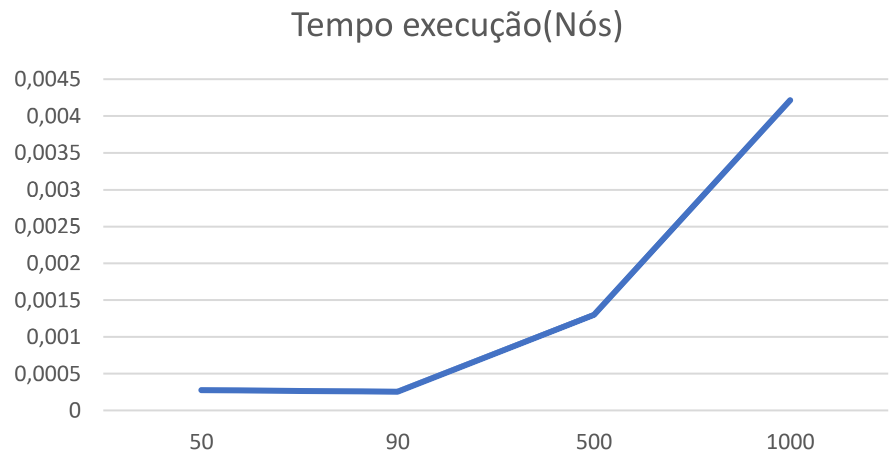
Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo (s)
1	50	146	0,0002747
7	90	257	0,0002563
3	500	1432	0,001299
5	1000	7535	0,0042166
9	5000	49487	0,0196982

```
Please insert the source station: :1

Please insert the destination station: :50

Maximum capacity: 19
```



Cenário 1: Grupos não se separam

Formalização 1.2:

Pretende-se implementar um algoritmo que não só satisfaça a premissa anterior, mas também que minimize o número de transbordos, sem privilegiar um dos critérios relativamente ao outro.

Função objetivo: Max C = Min(C_a) ∧ A ∈ {o, .., d} ∧ Min T, sendo C a capacidade e A uma aresta do grafo.

A capacidade do percurso (Max C), é neste caso o minimo das capacidades dos ramos que constituem o percurso (com maior mínimo). Já o número de transbordos (T) refere-se ao número de transbordos possiveis desde a origem ao destino.

Restrições: C = [0, + MP[= {C ∈ IR: 0 < C < MP}

T = [0, + número de nós(NN) [= {T ∈ IR: 0 < T < NN}

Input :

- Nó(inicial), Ni ∈ [1, tamanho_grafo] (ni , ni ∈ N)
- Nó(final), Nf ∈ [1, tamanho_grafo] (nf , nf ∈ N)
- Tamanho do grupo , Tg ∈ [1, Tg] (Tg , Tg ∈ N)

Output:

- Fluxo máximo entre os nós de input.
- Caminhos possíveis entre os nós de input satisfazendo a condição pedida.

Algoritmos utilizados e complexidades:

Algoritmo	Temporal	Espacial
1.2 - Caminhos de capacidade máxima (adaptação do Algoritmo de Dijkstra) através de lista de adjacências usando uma fila de prioridade suportada por um heap máximo.	O((|V| + |E|) log2 |V|)	O(1)
Algoritmo de Edmunds-Karp.		

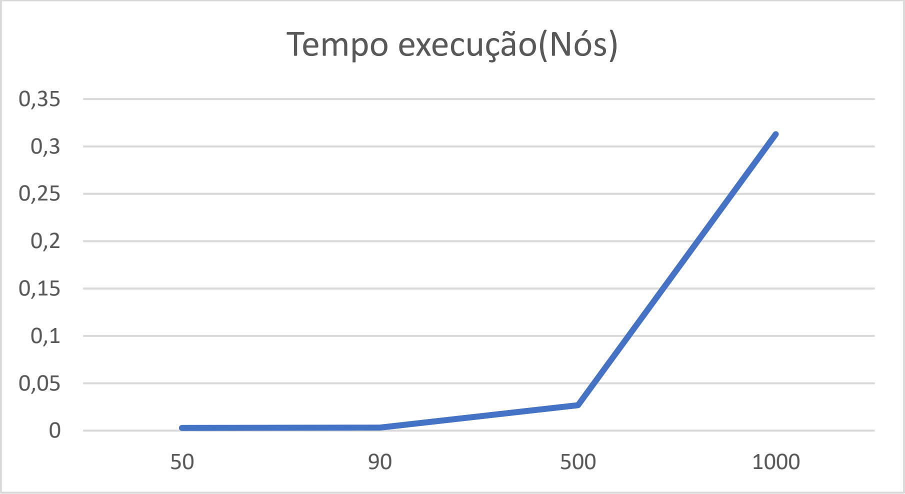
Resultados da avaliação empírica

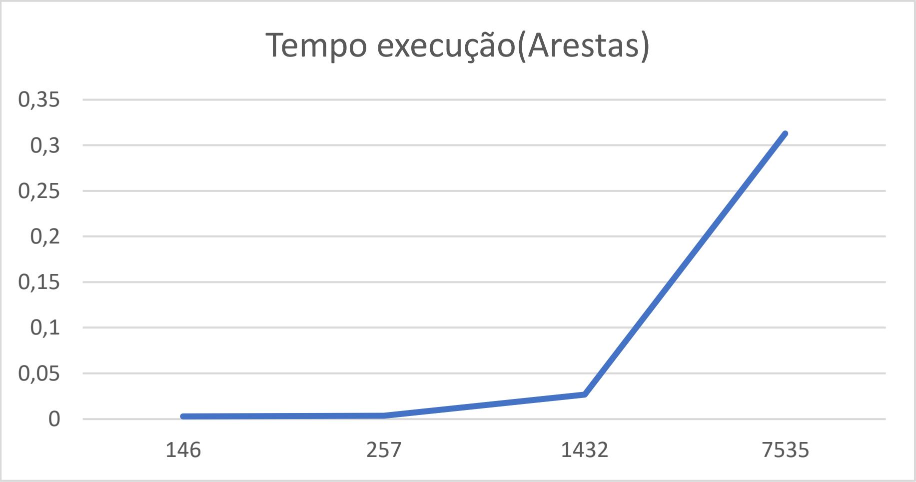
Grafo	Nos	Arestas	Tempo (s)
1	50	146	0,002912
7	90	257	0,003436
3	500	1432	0,02682
5	1000	7535	0,312884
9	5000	49487	18,1339

```
Scenario 1.2
1 - View Path
0 - Exit
:1

Maximum capacity 3, stops 5
1 -> 8 -> 46 -> 12 -> 50

Maximum capacity 5, stops 7
1 -> 6 -> 10 -> 36 -> 17 -> 39 -> 50
```





Cenário 2: Grupos que se podem separar

Formalização 2.1 :

Pretende-se determinar um encaminhamento para um grupo, dada a sua dimensão.

Função objetivo: $C = Min(\Sigma C_a(u, v)) \wedge (u \in S \wedge V \in T) \wedge a \in \{o, .., d\} \wedge s = \Sigma s(o, v) = \Sigma s(v, d) \wedge v \in \{o, .., d\} \wedge C >= s$, sendo C a capacidade máxima e S o tamanho do grupo dado pelo utilizador.

Restrições: $C \in [0,\text{maior peso da aresta do grafo}]$, $S \in [0,+\infty[$.

Input :

- Nó(inicial), $Ni \in [1, tamanho\_grafo]$ (ni , $ni \in N$)
- Nó(final), $Nf \in [1, tamanho\_grafo]$ (nf , $nf \in N$)
- Tamanho do grupo , $s \in [1, ,+\infty[$ (s , $s \in N$)

Output:

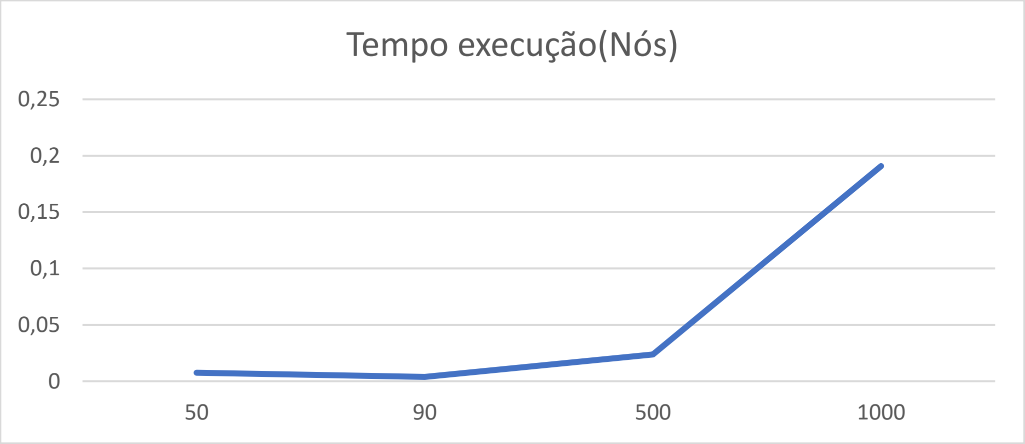
- Caminho possível para o grupo se deslocar de Ni a Nf, sendo que este caminho pode dividir-se em subcaminhos.

Algoritmos utilizados e comlexidades:

Algoritmo	Temporal	Espacial
2.1 – Algoritmo de Edmonds-Karp para a resolução do problema de fluxo máximo em uma rede de fluxo.	$O(\|V\|\|E\|^2)$	$O(\|V\|^2)$

Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo (s)
1	50	146	0.0077845
7	90	257	0.0040282
3	500	1432	0.0239415
5	1000	7535	0.190828
9	5000	49487	5.51315




```
Insert origin :1

Insert destination :50

Insert group size :15


Scenario 2.1:
1 - View Path
2 - Check when the group will get back together at the destination (at least) (2.4)
3 - Check the group's maximum waiting time during the path and when that occurs (2.5)
0 - Exit
:1

The path:
1 -> 299
1 -> 399
132 -> 903
```

Cenário 2: Grupos se podem separar

Formalização 2.2 :

Pretende-se corrigir um encaminhamento, se necessário, para que a dimensão do grupo possa aumentar de um número de unidades dado.

Função objetivo: $((ns > C) \rightarrow (C >= ns \land C = Min(\Sigma\ Cl(u, v)) \land (u \in S \land v \in T) \land l \in \{o, .., d\} \land ns = \Sigma\ ns(o, v) = \Sigma\ ns(v, d) \land v \in \{o, .., d\}\ )) \land (\neg(ns > C) \rightarrow (C = C))$.

Restrições:

- Capacidade inicial + Np <= C Max (o,d), sendo O a origem e D o destino

Input :

- Número de pessoas a aumentar ao grupo do algoritmo 2.1 , Np ∈ [1, tamanho_grafo] (np , np ∈ N)
- Caminho percorrido pelo algoritmo 2.1, Conjunto de arestas percorridas.

Output:

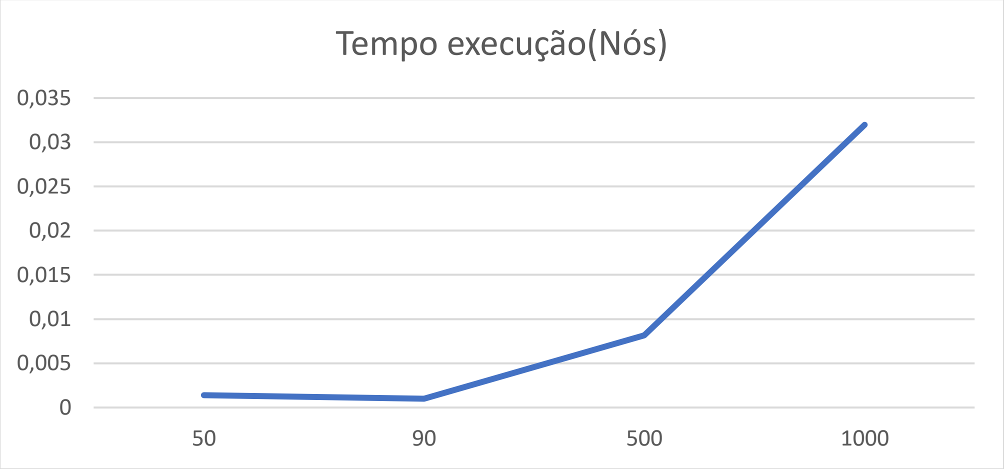
- Caminho percorrido pelo grupo após se adicionar Np pessoas.

Algoritmos utilizados e comlexidades:

Algoritmo	Temporal	Espacial
2.2 – Algoritmo de Edmonds-Karp para a resolução do problema de fluxo máximo em uma rede de fluxo.	$O(\|V\|\|E\|^2)$	$O(\|V\|^2)$

Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo (s)
1	50	146	0.0013877
7	90	257	0.0009936
3	500	1432	0.0081606
5	1000	7535	0.0319916
9	5000	49487	1.14




```
Insert group size increase for last the path of Scenario 2.1:6

Path capacity increased successfully

Scenario 2.2:
1 - View Path
2 - Check when the group will get back together at the destination (at least) (2.4)
3 - Check the group's maximum waiting time during the path and when that occurs (2.5)
0 - Exit
```

Cenário 2: Grupos se podem separar

Formalização 2.3 :

Pretende-se determinar a dimensão máxima do grupo e um encaminhamento. Como tal o objetivo prende-se em determinar o fluxo máximo de um ponto a outro.

Função objetivo: Max C = Min($\Sigma\ C_a(u, v) \wedge (u \in S \wedge v \in T) \wedge a \in \{o, .., d\} \wedge v \in \{o, .., d\} \wedge C = \Sigma\ C(o, v) = \Sigma\ C(v, d)$,

sendo C a capacidade.

A capacidade máxima desde a origem ao destino, é neste caso a capacidade minima de todos so subcaminhos do percurso.

Input :

- Nó(inicial), Ni ∈ [1, tamanho_grafo] (ni , ni ∈ N)
- Nó(final), Nf ∈ [1, tamanho_grafo] (nf , nf ∈ N)

Output:

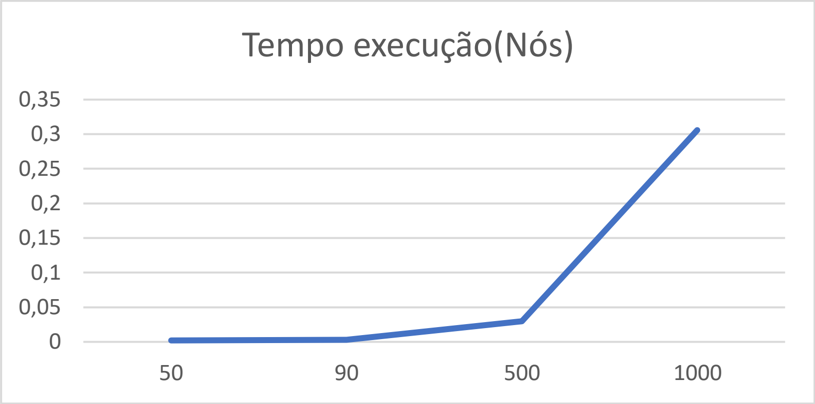
- Fluxo máximo entre os Nós de input

Algoritmos utilizados e complexidades:

Algoritmo	Temporal	Espacial
2.3 – Algoritmo de Edmonds-Karp para a resolução do problema de fluxo máximo em uma rede de fluxo.	$O(\|V\|\|E\|^2)$	$O(\|V\|^2)$

Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo(s)
1	50	146	0.0018327
7	90	257	0.0031219
3	500	1432	0.0296593
5	1000	7535	0.305787
9	5000	49487	17.7722




```
Insert origin :1

Insert destination :50

The maximum capacity of the path is 83

Scenario 2.3:
1 - View Path
2 - Check when the group will get back together at the destination (at least) (2.4)
3 - Check the group's maximum waiting time during the path and when that occurs (2.5)
0 - Exit
```

Cenário 2: Grupos que se podem separar

Formalização 2.4 :

Pretende-se partindo de um encaminhamento que constitui um grafo acíclico, determinar quando é que o grupo se reuniria novamente no destino, no mínimo.

Função objetivo: Min $T = \Sigma D_a(o, d)$, sendo $D_a(o, d)$ o tempo que se demora a percorrer um caminho que vai da origem o ao destino d.

O tempo mínimo desde a origem ao destino, é neste caso, o mínimo dos tempos que cada caminho acíclico leva a chegar de uma origem o a um destino d.

Input :

- Caminho retornado pelo algoritmo 2.1 ou 2.3. Conjunto de arestas percorridas desde o nó inicial ao nó destino.

Output :

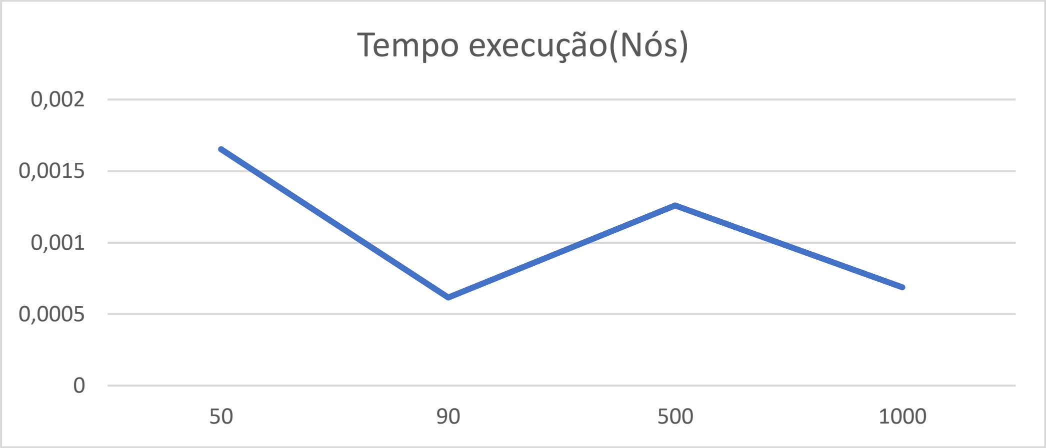
- Tempo até que o grupo se volte a encontrar totalmente $Tf \in [0, +\infty[$ ($Tf \in \mathbb{N}$)

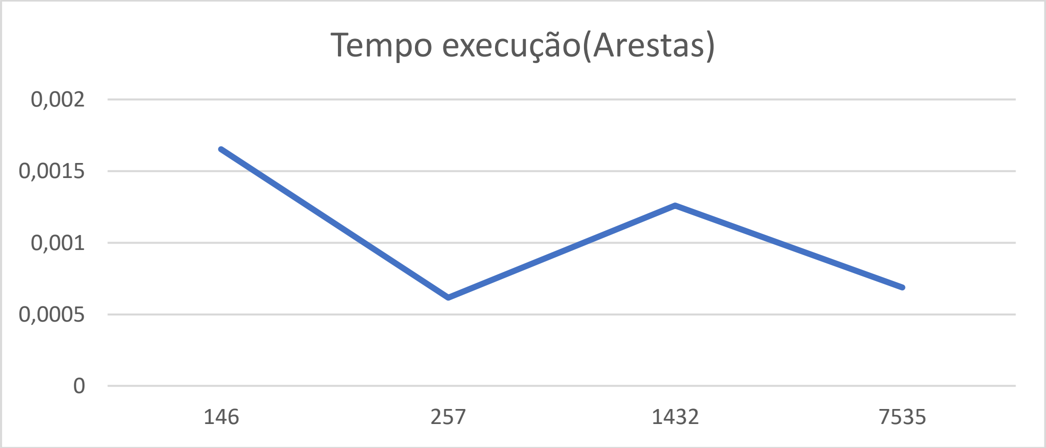
Algoritmos utilizados e complexidades:

Algoritmo	Temporal	Espacial
2.4 - Método do Caminho Crítico (arco-atividade) – earliest start, baseado no algoritmo de Edmonds-Karp.	O(|V|*Adj|V| + Log(V))	O(V)

Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo (s)
1	50	146	0,0016531
7	90	257	0,0006156
3	500	1432	0,0012596
5	1000	7535	0,0006884
9	5000	49487	0,0011288






```
Scenario 2.4:
Group will get back together at destination in 135 hour(s) (at least)
0 - Exit
:
```

Cenário 2: Grupos que se podem separar

Formalização 2.5 :

Pretende-se nas condições anteriores, admitindo que os elementos que saem de um mesmo local partem desse local à mesma hora (e o mais cedo possível), indicar o tempo máximo de espera e os locais em que haveria elementos que esperam esse tempo.

Função objetivo Para todo {w≠1 ∈ G.Estações} fazer

Para todo {(v , w) ∈ G.A} fazer

Tm[w] = max(ES[w] – ES[v] – tempo((v,w))

Input :

- Caminho retornado pelo algoritmo 2.1 ou 2.3. Conjunto de arestas percorridas desde o nó inicial ao nó destino.
- Conjunto dos tempos mínimos de espera nas diferentes estações

Output:

- Tempo de espera máximo entre transbordos
- Nós onde ocorre esse tempo de espera

Algoritmos utilizados e complexidades:

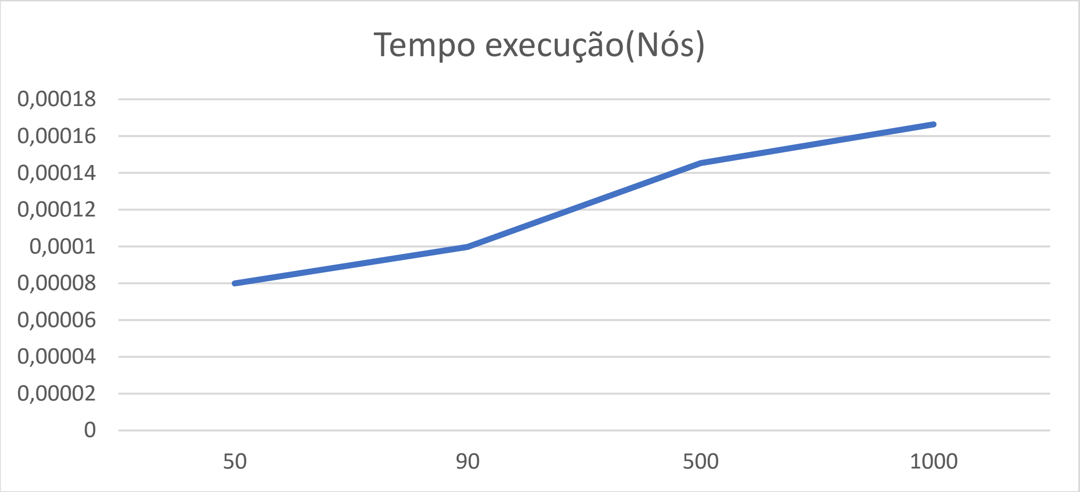
Algoritmo	Temporal	Espacial
2.5 - Para calcular o tempo de espera T numa estação D quando se vem de outra O, sendo G um grafo que contém um caminho composto por estações, A o conjunto de autocarros desse grafo e tempo(a ∈ A) o tempo que se demora nesse autocarro: T[D] = ES[D] - ES[O] - tempo((O,D)), sendo ES[V] o mais cedo possível que é possível partir de uma estação V, o que é dado pelo cenário 2.4.	O(2*(|V|*|E|) + Log(V))+	O(V)
Para calcular o tempo de espera máximo de espera Tm numa estação D: Para todo {(v , w) ∈ G.A | w = D} fazer Tm[D] = max(ES[D] – ES[v] – tempo((v,D)) Em seguida, para calcular o tempo de espera máximo num caminho C, basta repetir o algoritmo anterior para todas as estações {E : E ∈ C}.	O(|V|*|E| + Log(V))	

Resultados da avaliação empírica

Grafo	Nos	Arestas	Tempo (s)
1	50	146	0,0000799
7	90	257	0,0000997
3	500	1432	0,0001453
5	1000	7535	0,0001664
9	5000	49487	0,0018702

```
Scenario 2.5:
Max time people have to wait: 99
1 - See Stations where that happens
0 - Exit
:1

That happens in 1 stations
Stations nr: 50
```



User Interface (exemplos)

Menu Principal

```
Main Menu:
1 - First Scenario - groups that don't split
2 - Second Scenario - groups that can split
3 - Settings Menu
0 - Exit
```

Menu de Settings

```
Settings Menu:
1 - Change current DataSet (current: 01)
2 - Toggle DataSet type (B type: false)
0 - Exit
```

View Information

```
Maximum capacity: 5
:1

1 -> 6 -> 10 -> 36 -> 17 -> 39 -> 50
```

Principais Dificuldades

A principal dificuldade sentida pelo grupo foi a implementação dos três primeiros algoritmos do segundo cenário, levando em conta a dependência que os outros têm destes. Por motivos semelhantes, a organização da estrutura da nossa aplicação foi desafiante.

Esforço de cada elemento:

O grupo esforçou-se de igual forma, tendo sempre como objetivo desenvolver algoritmos o mais eficientemente possíveis. Desta forma a repartição do esforço de cada elemento é de 33.(3) %.