2017-18 Recurso

1

- a) Pode-se optar por uma estratégia gananciosa que optaria por escolher parar nas estações mais próximas da distância m, mas não com distância superior.

```
int d[n];
int min = m;
int print = 0;
for (int i = 1; i <= D/m; i++){ //Número de vezes que tem que parar
    for (int j = 0; j < n; j++){
        if (abs(d[j] - m*i) < min && d[j] < m){
            min = m*i - d[j];
            print = j + 1;
        }
    }
    cout << print << endl;
}
```

- b) Complexidade temporal: O(D/m * n) Compplexidade espacial: O(1)

O algoritmo poderá ser otimizado uma vez que muitos cálculos são repetidos para cada paragem. Poderia-se optar por uma estratégia de programação dinâmica onde se guardassem distâncias, evitando o cálculo no futuro.

2

- a)

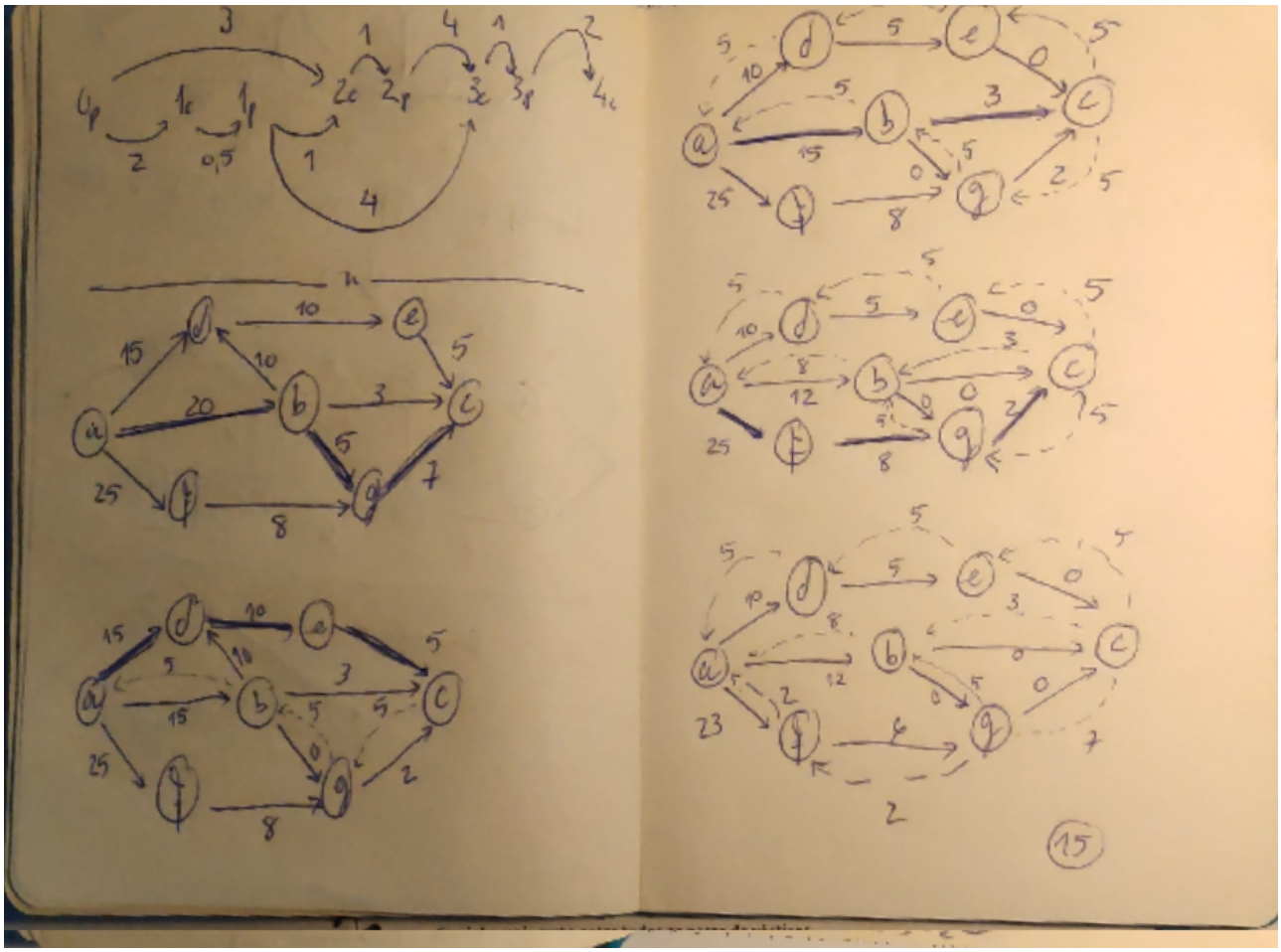
	0p	**1c**	**1p**	**2c**	**2p**	**3c**	**3p**	**4c**
0p	0	2		3				
1c	0	2	2,5	3				
1p	0	2	2,5	3		4		
2c	0	2	2,5	3	4	4		
2p	0	2	2,5	3	4	4		
3c	0	2	2,5	3	4	4	5	
3p	0	2	2,5	3	4	4	5	7
4c	0	2	2,5	3	4	4	5	7

- b)

Execução do algoritmo de Dijkstra: O((|V|+|E|*log(V)))

3

- a)



- b) É reduzida em 3V.
- c) É suportada pelo circuito original, uma vez que a corrente máxima(15) é superior ao valor indicado.

4

- a) Será possível formar uma equipa com um número de trabalhadores que já trabalharam em conjunto dois a dois superior a k?
- b) O problema é NP-completo, logo não resolúvel em tempo polinomial:
    - É NP, pois é possível verificar se uma equipa é composta por trabalhadores que já tinham trabalhado no mesmo projeto
    - É NP-difícil, pois o problema da (...) é redutível em tempo polinomial ao problema da formação de equipas.