

Problema dei Tubi

A short story

Luca Mazza Vasco Silva Pereira Sebastiano Piubellini

December 14, 2024

Table of Contents

- 1 Introduzione
- 2 Risoluzione del problema
- 3 Complessità
- 4 Conclusioni

Table of Contents

1 Introduzione

2 Risoluzione del problema

3 Complessità

4 Conclusioni

Introduzione

Problema di ottimizzazione di una fabbrica di cioccolato, manutenzione delle linee di trasporto del cioccolato.

- Giunti interconnettono i rami di produzione (*nodes*)
- Ogni giunto ha un tempo di manutenzione
- Giunti di ricambio a disposizione
- Cambiando un giunto si azzerà

Table of Contents

1 Introduzione

2 Risoluzione del problema

3 Complessità

4 Conclusioni

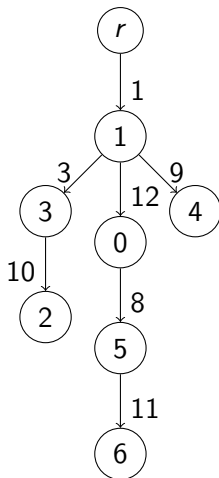
Risoluzione del problema

Ci siamo basati sul prototipo dell'algoritmo **DFS** per sviluppare l'algoritmo. Fin da subito, abbiamo capito che questo poteva essere una possibile soluzione del problema.

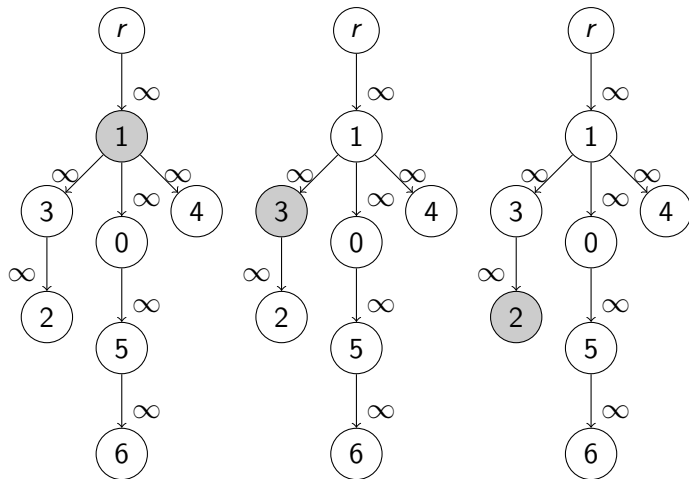
Funzionamento

DFS esplora un albero, partendo dalla radice, dando priorità all'esplorazione in profondità.

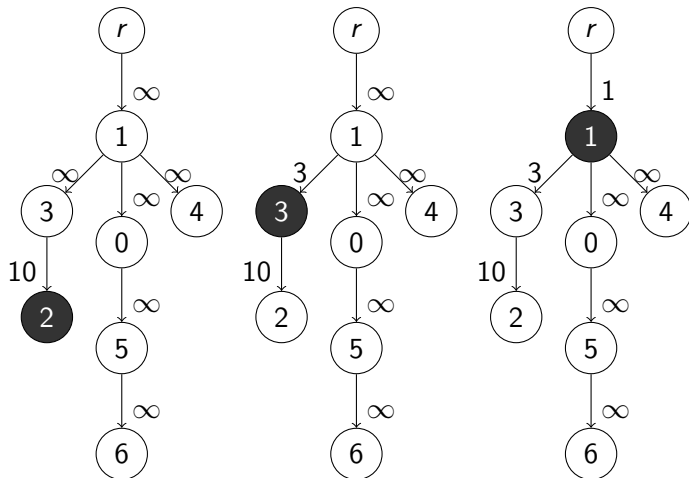
Risoluzione del problema



Visita in profondità



Riavvolgimento



Pseudocodice I

```
1: procedure MINIMIZE_TIME(node)  
2:   if child_count[node] = 0 then  
3:     mem[node][0]  $\leftarrow T$ [node]  
4:     for c  $\leftarrow$  1 to C do  
5:       mem[node][c]  $\leftarrow$  0  
6:     end for  
7:     return  
8:   end if
```

▷ Base case

Pseudocodice II

```
9:   for each child  $v$  of  $node$  in  $adj[node]$  do                                ▷ Recursive case
10:      MINIMIZE TIME( $v$ )
11:      Initialize  $cur[0 \dots C] \leftarrow \infty$ 
12:      for  $c \leftarrow 0$  to  $C$  do
13:         for  $x \leftarrow 0$  to  $c$  do
14:             $cur[c] \leftarrow \min(cur[c], \max(mem[v][x], mem[node][c - x]))$ 
15:         end for
16:      end for
17:       $mem[node][...] \leftarrow cur[...]$ 
18:   end for
19:   Initialize  $cur[0 \dots C]$ 
20:    $cur[0] \leftarrow T[node] + mem[node][0]$ 
```

Pseudocodice III

```
21:   for  $c \leftarrow 1$  to  $C$  do                                ▷ Wrap-up
22:        $\text{cur}[c] \leftarrow \min(T[\text{node}] + \text{mem}[\text{node}][c], \text{mem}[\text{node}][c - 1])$ 
23:   end for
24:    $\text{mem}[\text{node}][...] \leftarrow \text{cur}[...]$ 
25: end procedure
```

Risoluzione del problema

Una volta visitati tutti i nodi l'algoritmo prova, per ogni ricambio, ad applicare c ricambi e $(c - x)$ per il resto del sistema. Viene poi aggiornato il valore attuale calcolato per c ricambi che minimizza il massimo valore tra:

- Il risultato del sottoalbero radicato in v con x ricambi;
- Il risultato del nodo corrente e degli altri sottoalberi con $c - x$ ricambi.

$$cur(c) = \min(cur(c), \max(S_v(x), S_u(c - x)))$$

Table of Contents

1 Introduzione

2 Risoluzione del problema

3 Complessità

4 Conclusioni

Complessità

- N : giunti da riparare, C : giunti sostituibili
- Complessità temporale: $O(N * C^2)$
- Complessità spaziale: $O(N * C)$
- Dati calcolati da una media di 10 test per ogni task

Complessità - memoria occupata




memoria occupata.png

- La memoria occupata cresce proporzionalmente a numero giunti (N) e giunti sostituibili (C)

Figure: Utilizzo memoria

Complessità - tempo di esecuzione



tempo cpu.png

Figure: Tempo CPU

Table of Contents

1 Introduzione

2 Risoluzione del problema

3 Complessità

4 Conclusioni

Sample frame title

In this slide, some important text will be **highlighted** because it's important. Please, don't abuse it.

Remark

Sample text

Important

Sample text in red box

Examples

Sample text in green box. The title of the block is "Examples".