Algoritmos gulosos (*greedy*)

CLRS Cap 16 Greedy Algorithms

Problema dos intervalos disjuntos

Problema: Dados intervalos semi-abertos $[s[1], f[1]), \ldots, [s[n], f[n])$ encontrar uma coleção máxima de intervalos disjuntos dois a dois.

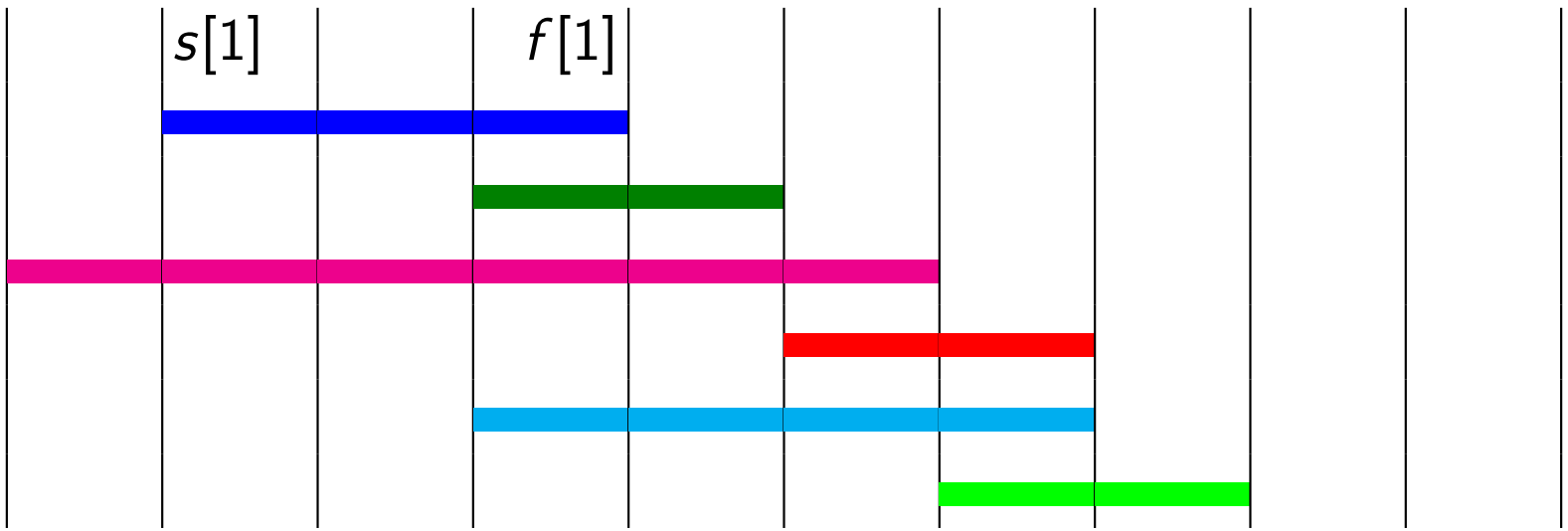
Solução é um subconjunto A de $\{1, \ldots, n\}$.

Problema dos intervalos disjuntos

Problema: Dados intervalos semi-abertos $[s[1], f[1]), \ldots, [s[n], f[n])$ encontrar uma coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Exemplo:

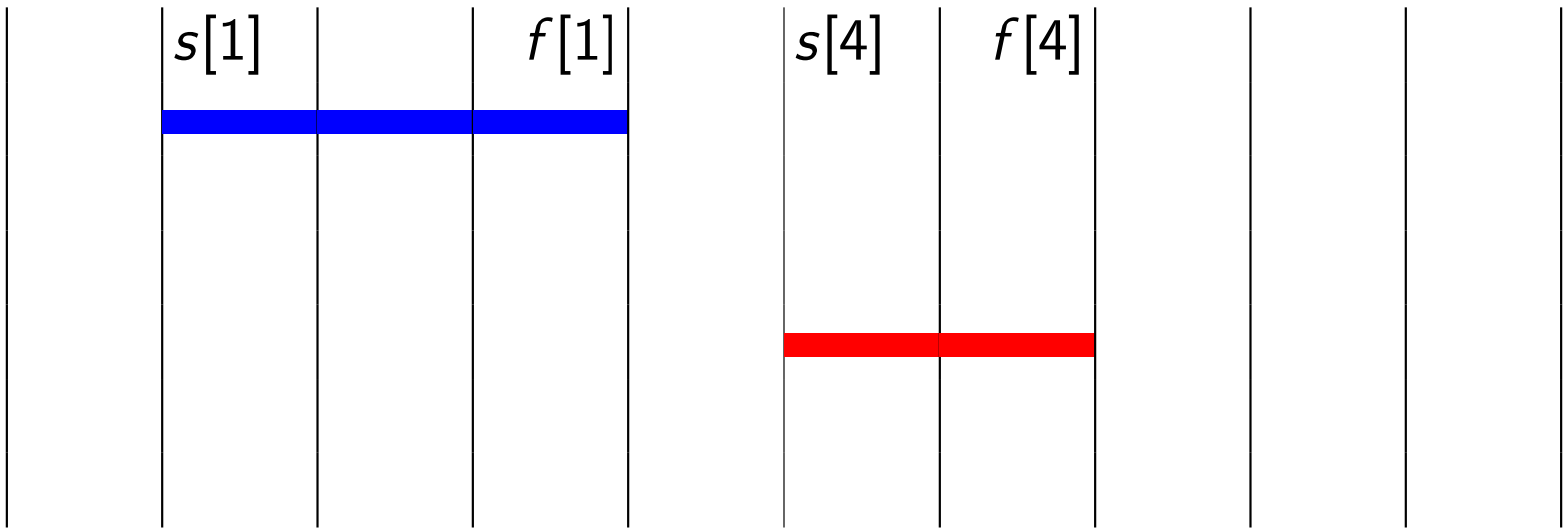


Problema dos intervalos disjuntos

Problema: Dados intervalos semi-abertos $[s[1], f[1]), \ldots, [s[n], f[n])$ encontrar uma coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Solução:

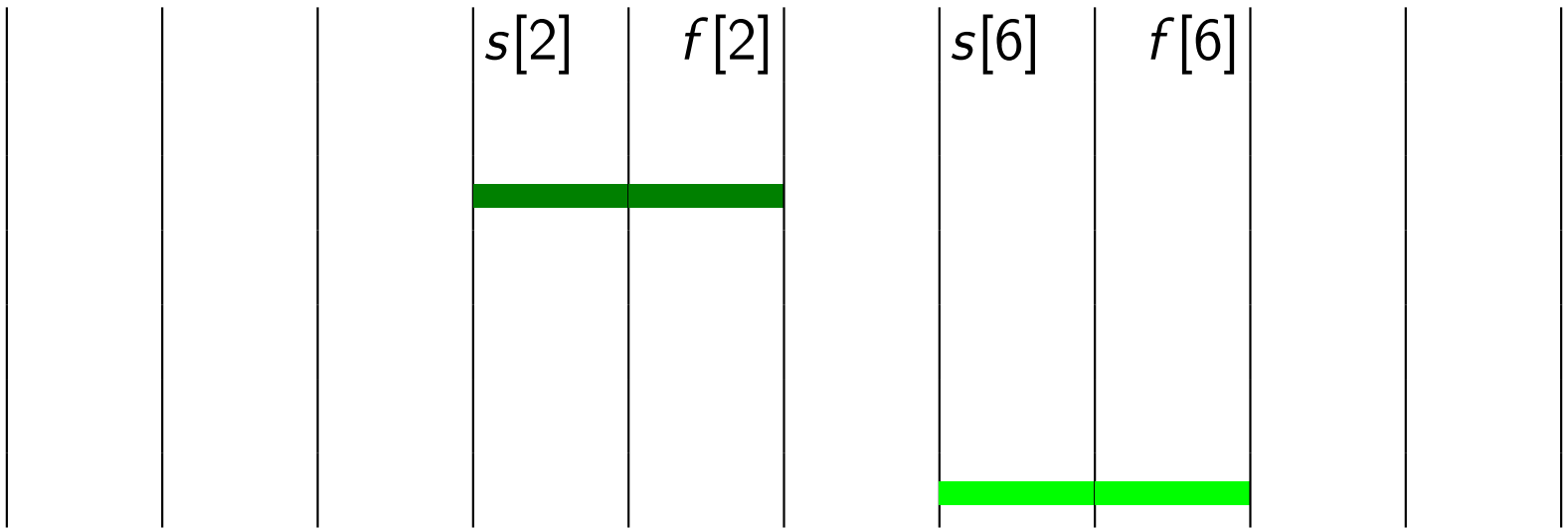


Problema dos intervalos disjuntos

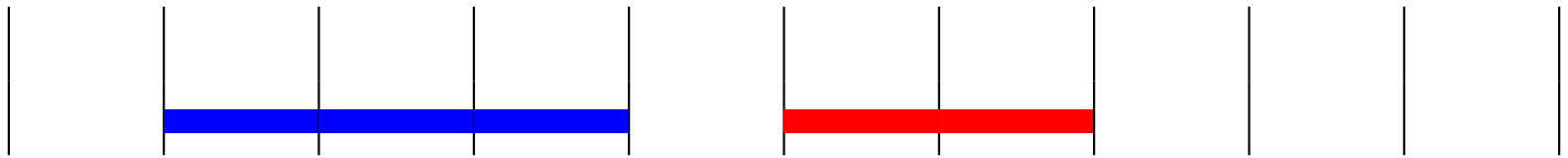
Problema: Dados intervalos semi-abertos $[s[1], f[1]), \ldots, [s[n], f[n])$ encontrar uma coleção máxima de intervalos disjuntos dois a dois.

Solução é um subconjunto A de $\{1, \ldots, n\}$.

Solução:



Motivação



Se cada intervalo é uma "atividade", queremos coleção disjunta máxima de atividades compatíveis (i e j são compatíveis se $f[i] \leq s[j]$ ou $f[j] \leq s[i]$).

Nome no CLRS: Activity Selection Problem

Subestrutura ótima

Intervalos $S := \{1, \ldots, n\}$

Suponha que

A é coleção máxima de intervalos disjuntos de S.

Se $i \notin A$

então

Subestrutura ótima

Intervalos $S := \{1, \ldots, n\}$

Suponha que

A é coleção máxima de intervalos disjuntos de S.

Se $i \notin A$

então A é coleção máxima de intervalos disjuntos de $S \setminus \{i\}$.

senão

Subestrutura ótima

Intervalos $S := \{1, \ldots, n\}$

Suponha que

A é coleção máxima de intervalos disjuntos de S.

Se $i \notin A$

então A é coleção máxima de intervalos disjuntos de $S \setminus \{i\}$.

senão $A \setminus \{i\}$ é coleção máxima de intervalos disjuntos de $S \setminus \{k \in S : k \text{ é incompatível com } i\}$.

Demonstre a propriedade.

Subestrutura ótima II

Intervalos $S := \{1, \ldots, n\}$

Suponha que

A é coleção máxima de intervalos disjunto de S.

Se $i \notin A$

então A é coleção máxima de intervalos disjuntos de $S \setminus \{i\}$.

senão suponha que i é o primeiro intervalo de A

Subestrutura ótima II

Intervalos $S := \{1, \ldots, n\}$

Suponha que

A é coleção máxima de intervalos disjunto de S.

Se $i \notin A$

então A é coleção máxima de intervalos disjuntos de $S \setminus \{i\}$.

senão suponha que i é o primeiro intervalo de A
$A \setminus \{i\}$ é coleção máxima de
intervalos disjuntos de $\{k \in S : s[k] \geq f[i]\}$.

$\{k \in S : s[k] \geq f[i]\}$ = todos intervalos "à direita" de "i".

Demonstre a propriedade.

Algoritmo de programação dinâmica

Suponha $s[1] \leq s[2] \leq \cdots \leq s[n]$.

$t[i] =$ tamanho de uma subcoleção disjunta máxima de $\{i, \ldots, n\}$

Algoritmo de programação dinâmica

Suponha $s[1] \leq s[2] \leq \cdots \leq s[n]$.

$t[i] =$ tamanho de uma subcoleção disjunta máxima de $\{i, \ldots, n\}$

$$t[n] = 1$$

$$t[i] = \max\{t[i+1], 1 + t[k]\} \quad \text{para } i = 1, \ldots, n-1,$$

onde k é o menor índice tal que $s[k] \geq f[i]$.

Algoritmo de programação dinâmica

DYNAMIC-ACTIVITY-SELECTOR (s, f, n)

0 ordene s e f de forma que $s[1] \leq s[2] \leq \cdots \leq s[n]$

1 $A[n+1] \leftarrow \emptyset$
2 **para** $i \leftarrow n$ **decrescendo até** 1 **faça**
3 $\quad A[i] \leftarrow A[i+1]$
4 $\quad k \leftarrow i+1$
5 $\quad$ **enquanto** $k \leq n$ e $s[k] < f[i]$ **faça**
6 $\quad\quad k \leftarrow k+1$
7 $\quad$ **se** $|A[i]| < 1 + |A[k]|$
8 $\quad\quad$ **então** $A[i] \leftarrow \{i\} \cup A[k]$
9 **devolva** $A[1]$

Consumo de tempo é $\Theta(n^2)$.

Conclusão

Invariante: na linha 2 vale que

(i0) $A[k]$ *é coleção disjunta máxima de* $\{k, \ldots, n\}$ *para* $k = i + 1, \ldots, n$.

O consumo de tempo do algoritmo DYNAMIC-ACTIVITY-SELECTOR é $\Theta(n^2)$.

Algoritmo guloso

Considere o problema da coleção máxima de intervalos disjuntos.

GULOSO (s, f, n)
1 $A \leftarrow \emptyset$
2 $R \leftarrow \{1, \ldots, n\}$
3 **enquanto** $R \neq \emptyset$ **faça**
4 escolha por um critério guloso um intervalo i de R
5 $A \leftarrow A \cup \{i\}$
6 $R \leftarrow R \setminus \{j \in R : j \text{ intersecta } i\}$
7 **devolva** A

Claro que A é uma coleção de intervalos compatíveis.

Qual critério usamos para escolher um intervalo de R?

Possíveis critérios gulosos

- escolher o intervalo i em R com menor $s[i]$;

Possíveis critérios gulosos

- escolher o intervalo i em R com menor $s[i]$;
- escolher o intervalo i em R com menor $f[i] - s[i]$;

Possíveis critérios gulosos

- escolher o intervalo i em R com menor $s[i]$;
- escolher o intervalo i em R com menor $f[i] - s[i]$;
- escolher o intervalo i em R com menor $f[i]$;

Possíveis critérios gulosos

- escolher o intervalo i em R com menor $s[i]$;
- escolher o intervalo i em R com menor $f[i] - s[i]$;
- escolher o intervalo i em R com menor $f[i]$;
- escolher o intervalo i tal que

$$|\{j \in R : j \text{ intersecta } i\}|$$

é o menor possível;

Escolha gulosa

Intervalos $S := \{1, \ldots, n\}$

Se $f[i]$ é mínimo em S,

então EXISTE uma solução ótima A tal que $i \in A$.

Demonstre a propriedade.

Algoritmo guloso

Devolve uma coleção máxima de intervalos disjuntos dois a dois.

```
INTERVALOS-DISJUNTOS (s, f, n)
0  ordene s e f de forma que f[1] ≤ f[2] ≤ ··· ≤ f[n]
1  A ← {1}
2  i ← 1
3  para j ← 2 até n faça
4      se s[j] ≥ f[i]
5          então A ← A ∪ {j}
6                i ← j
7  devolva A
```

Consumo de tempo da linha 0 é $\Theta(n \lg n)$.
Consumo de tempo das linhas 1–7 é $\Theta(n)$.

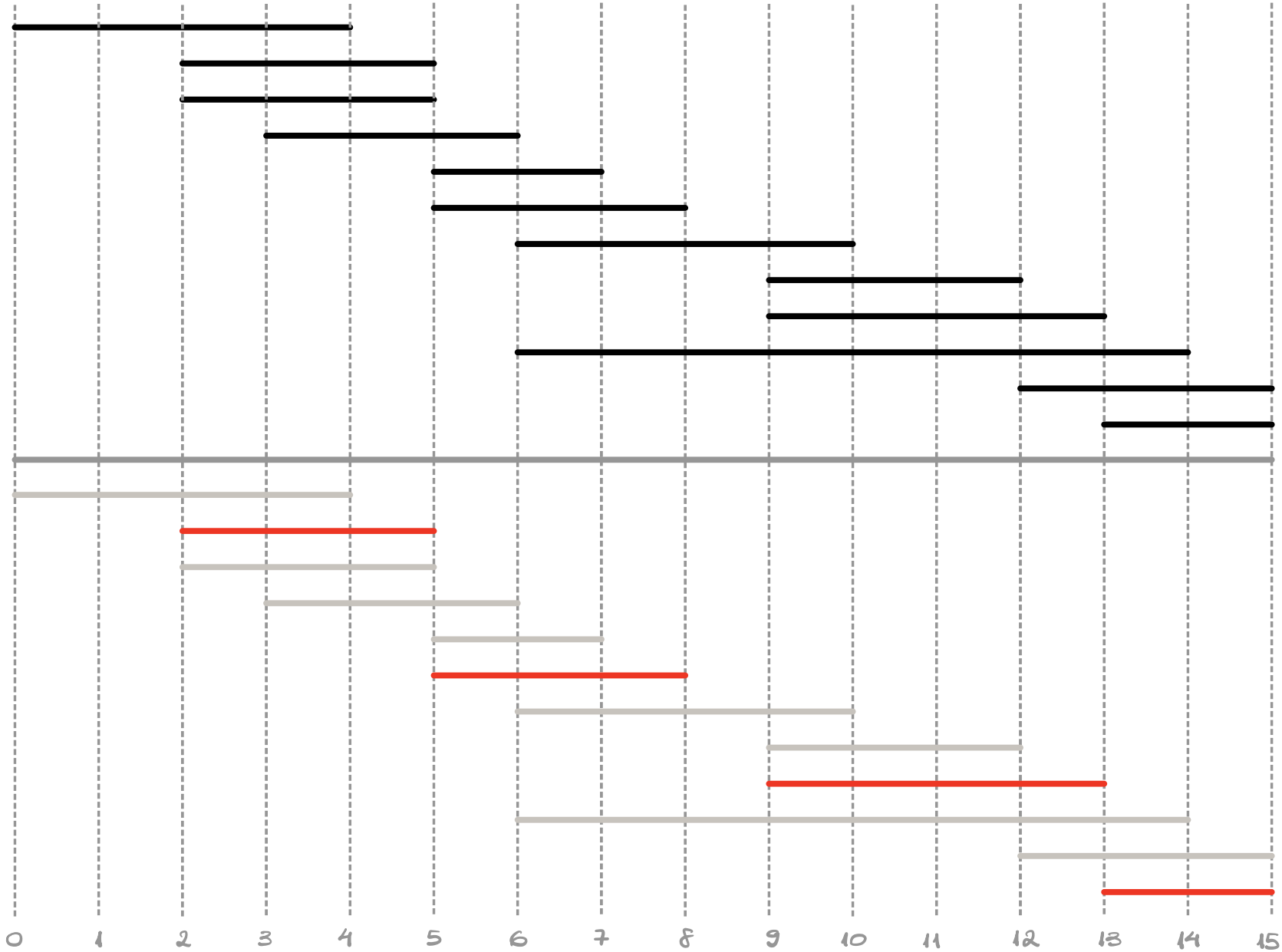
Conclusão

Na linha 3 vale que

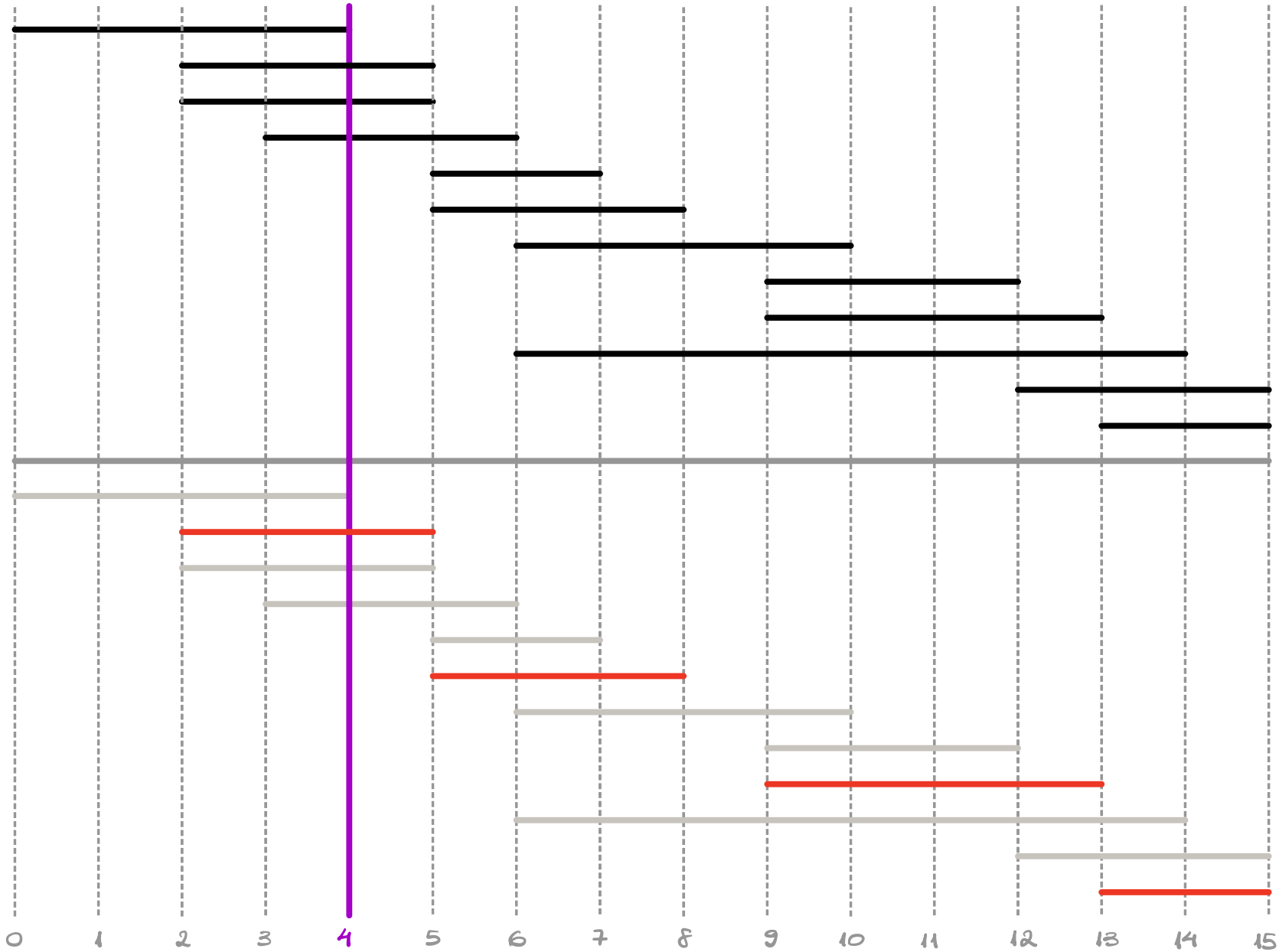
(i0) A é uma coleção máxima de intervalos disjuntos de $(s, f, j-1)$

O consumo de tempo do algoritmo INTERVALOS-DISJUNTOS é $\Theta(n \lg n)$.

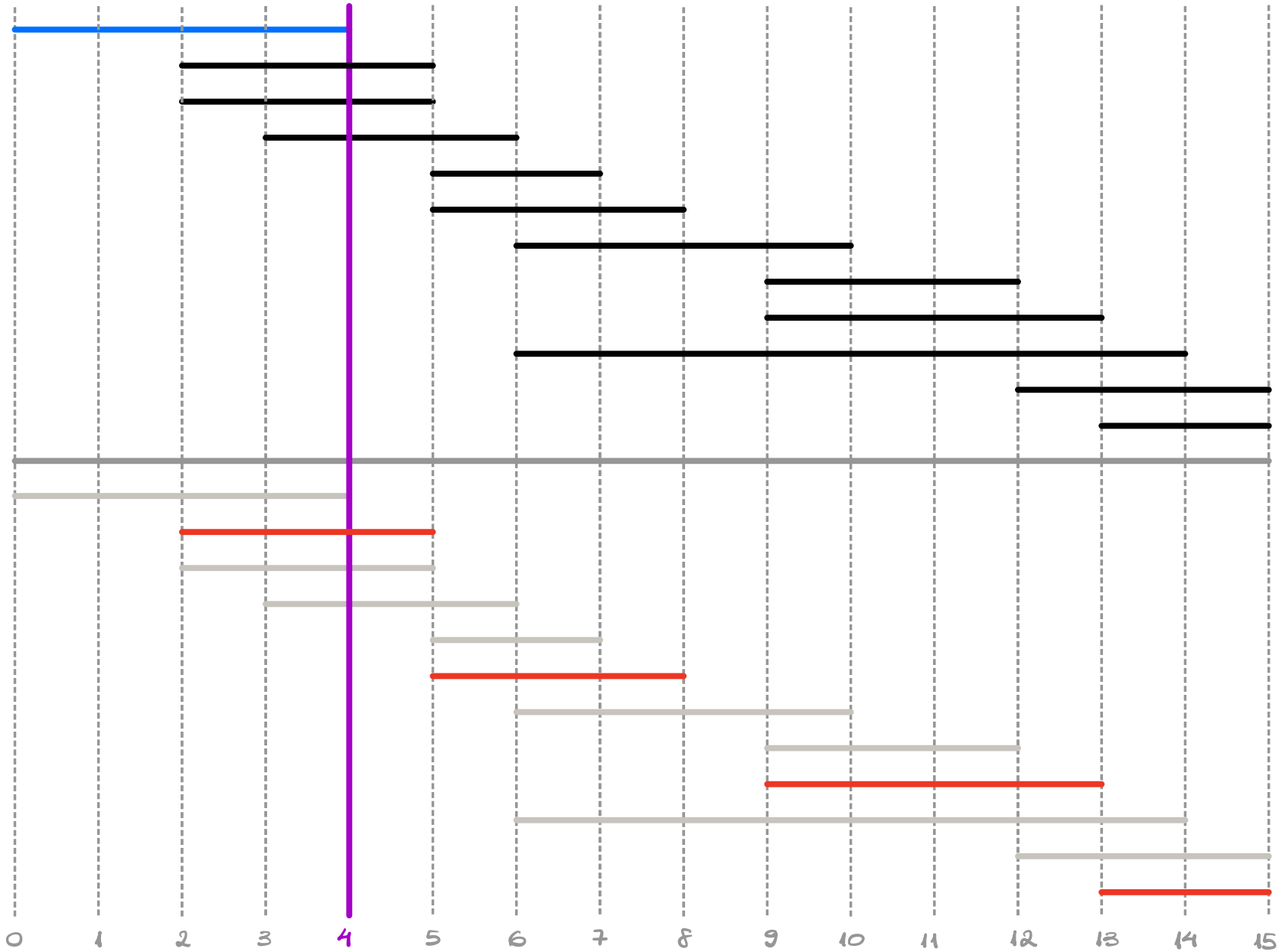
Escolha gulosa e concordância de soluções



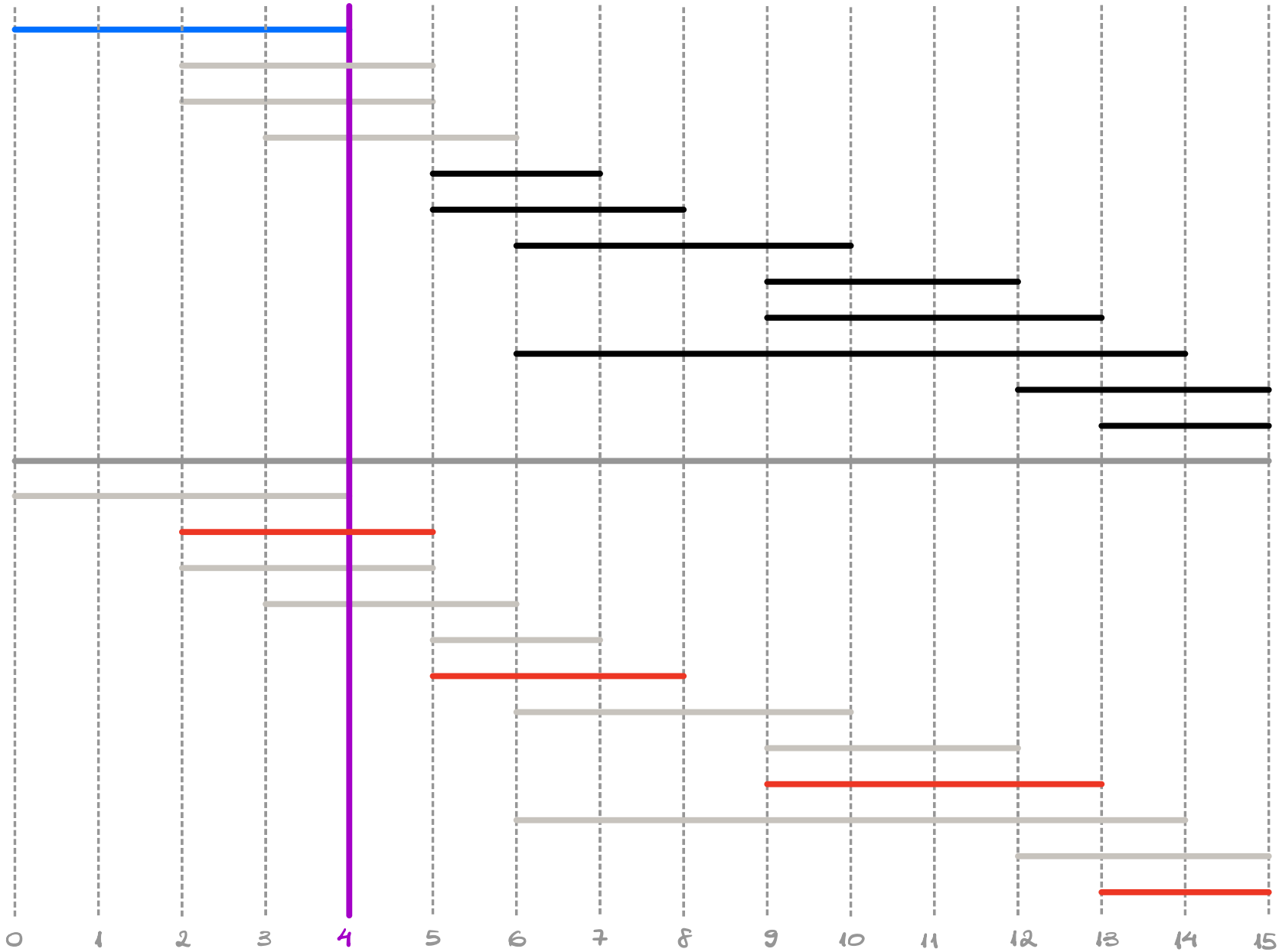
Escolha gulosa e concordância de soluções



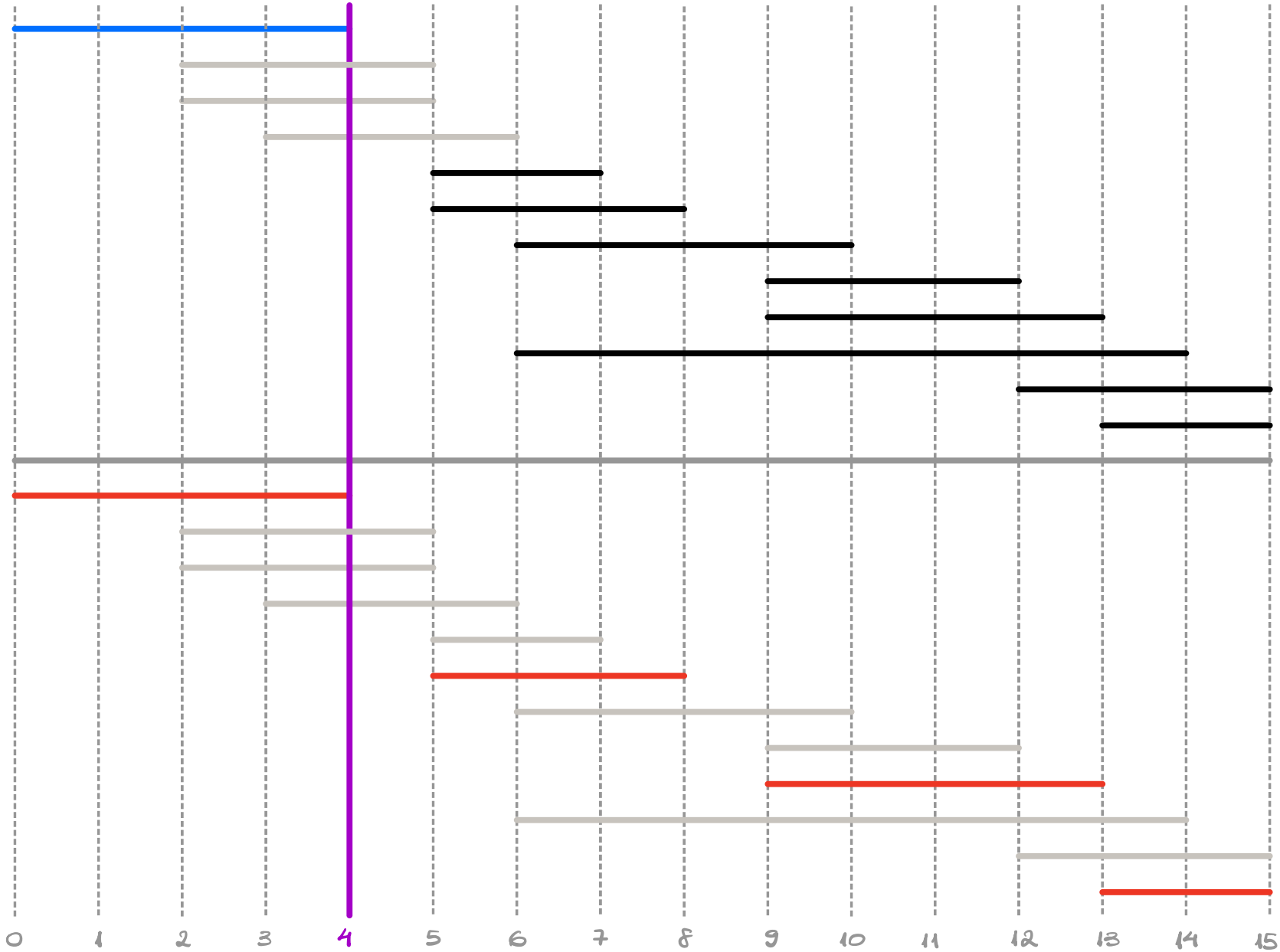
Escolha gulosa e concordância de soluções



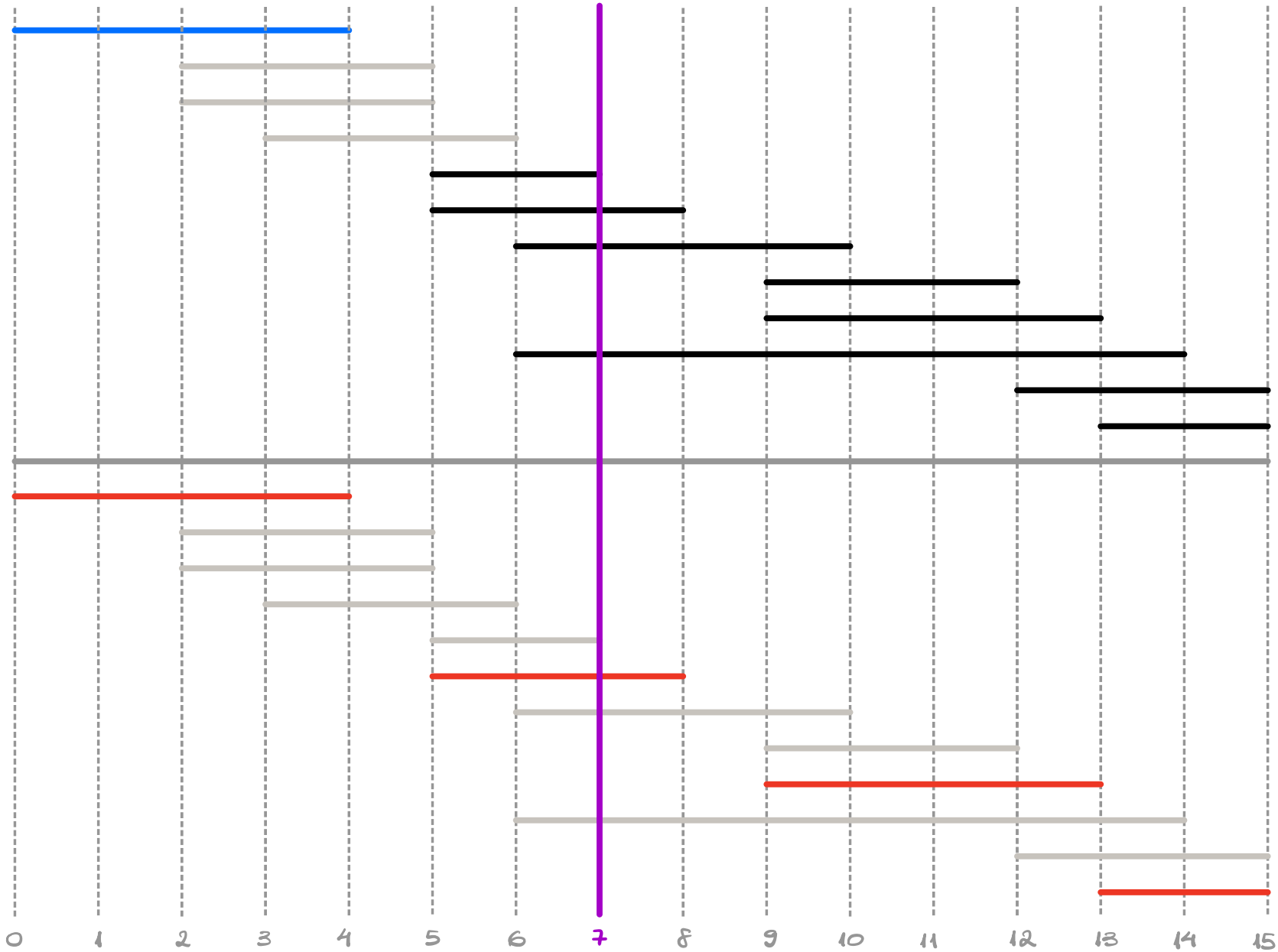
Escolha gulosa e concordância de soluções



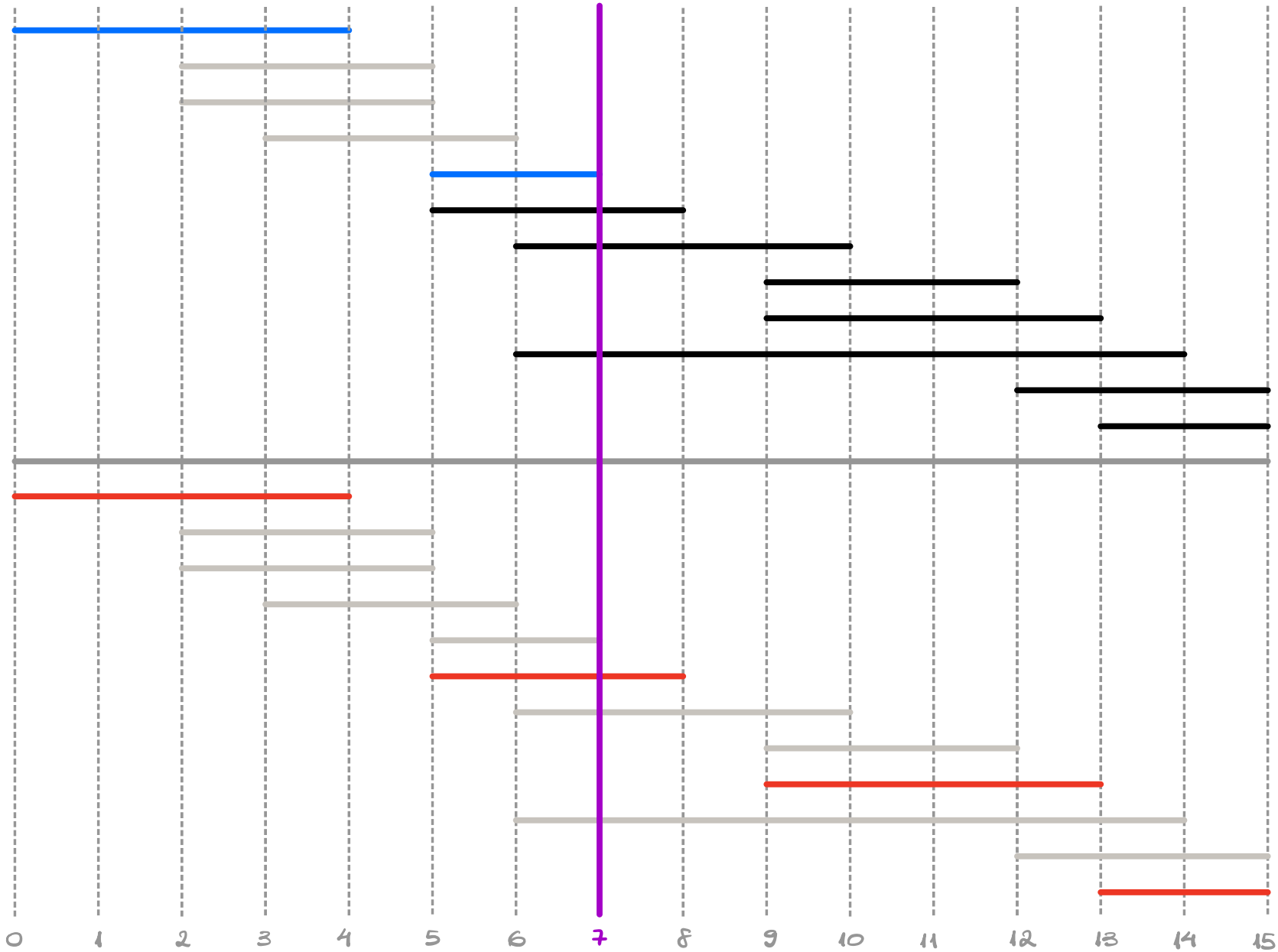
Escolha gulosa e concordância de soluções



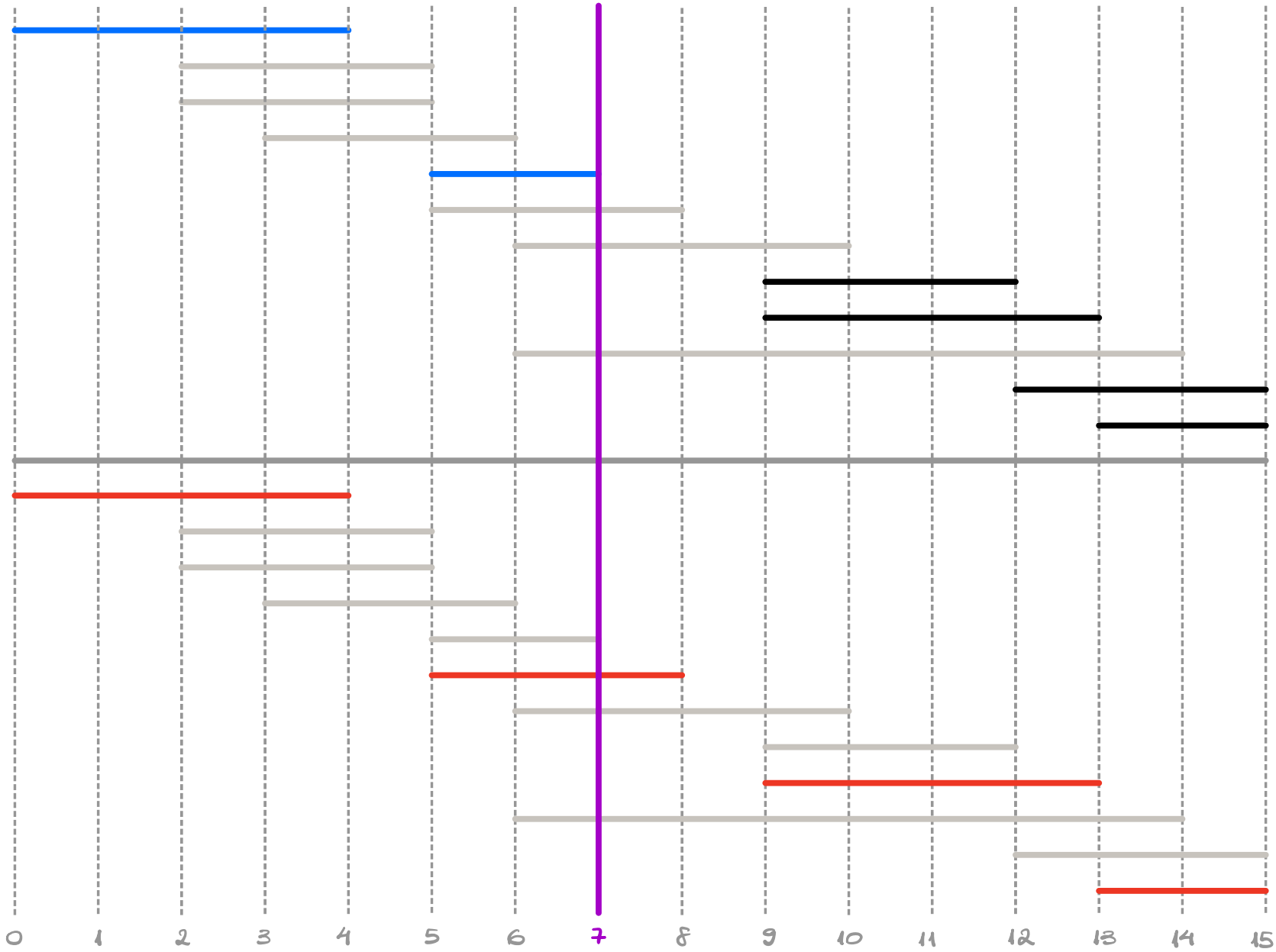
Escolha gulosa e concordância de soluções



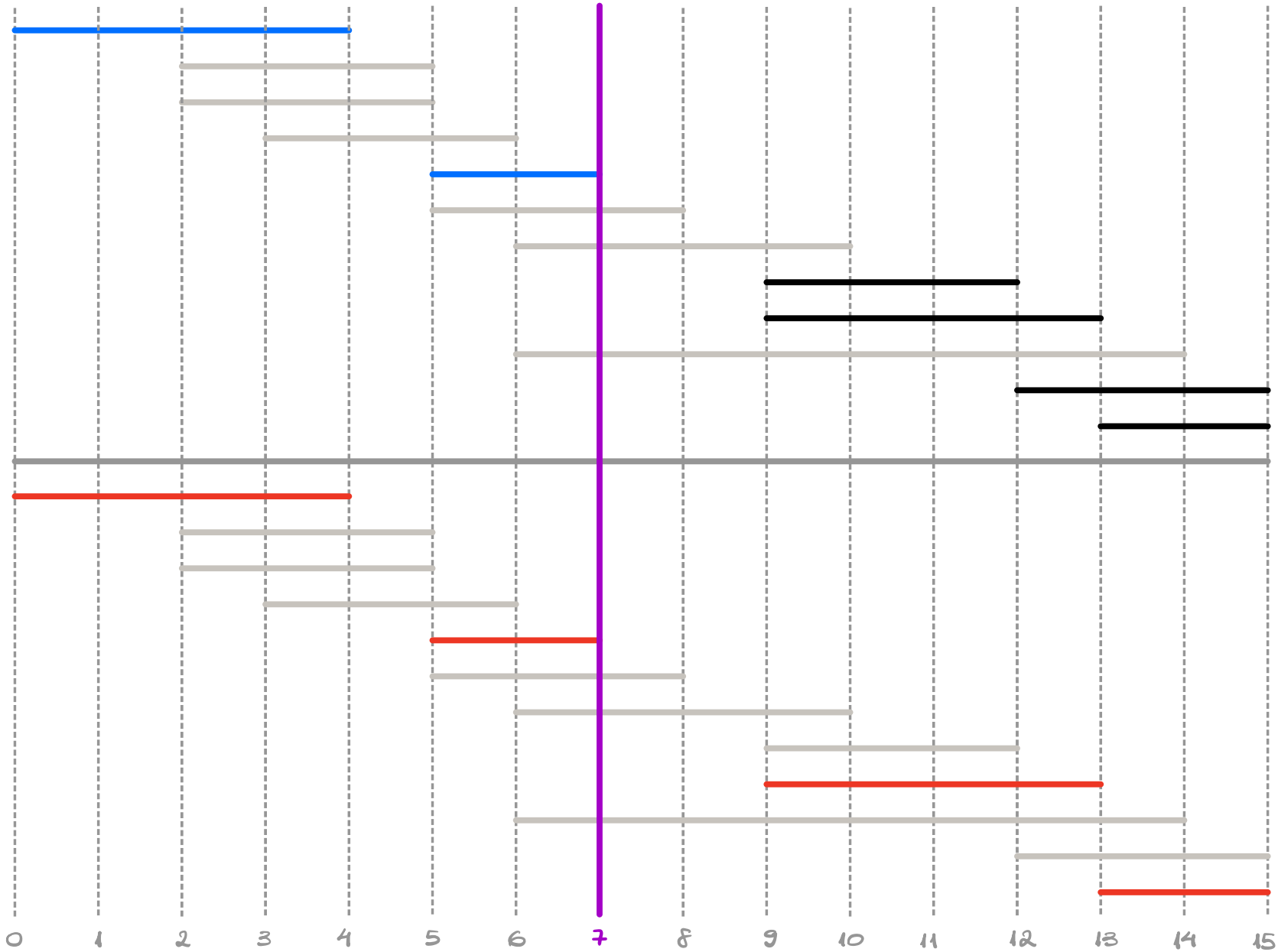
Escolha gulosa e concordância de soluções



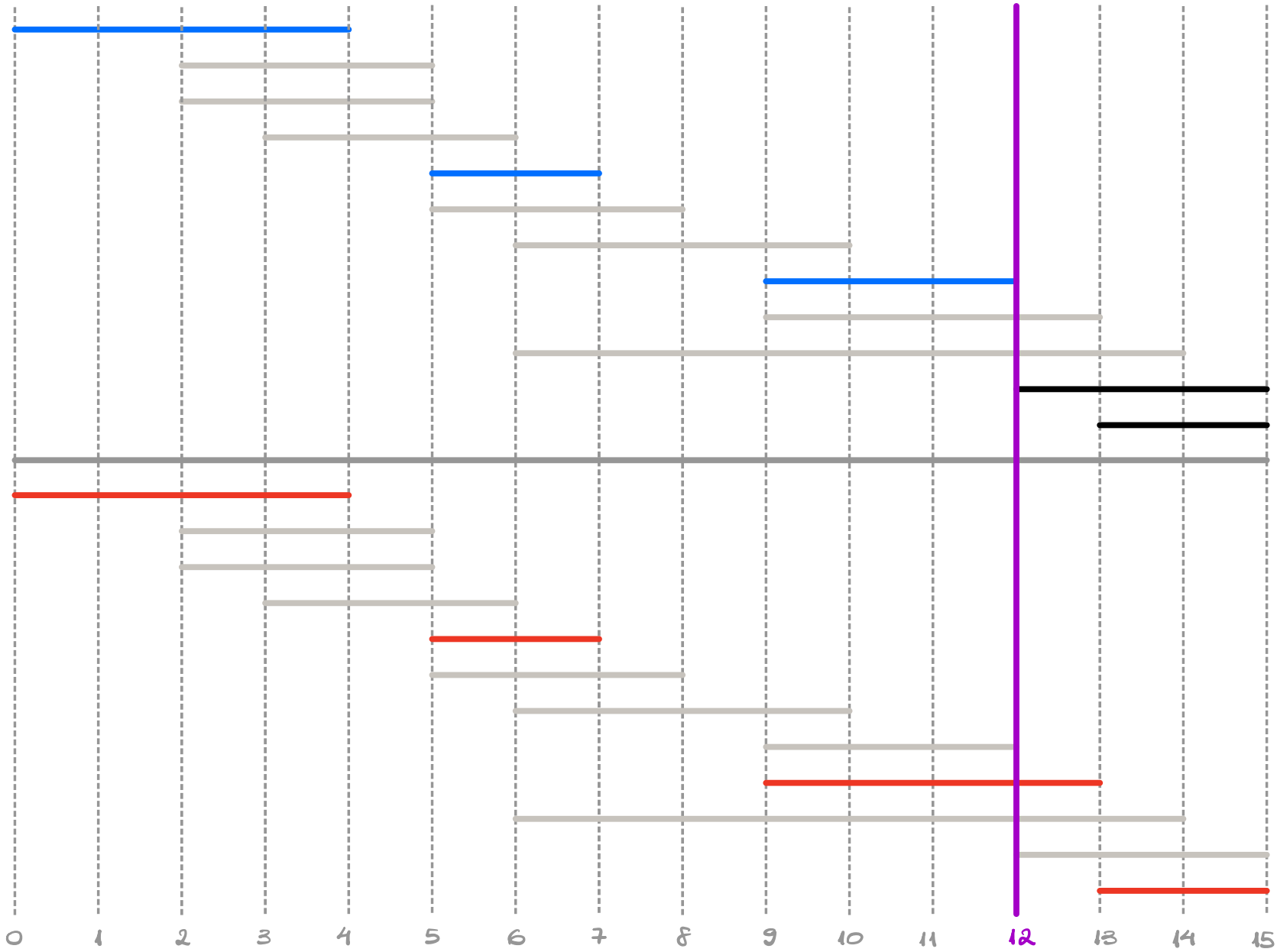
Escolha gulosa e concordância de soluções



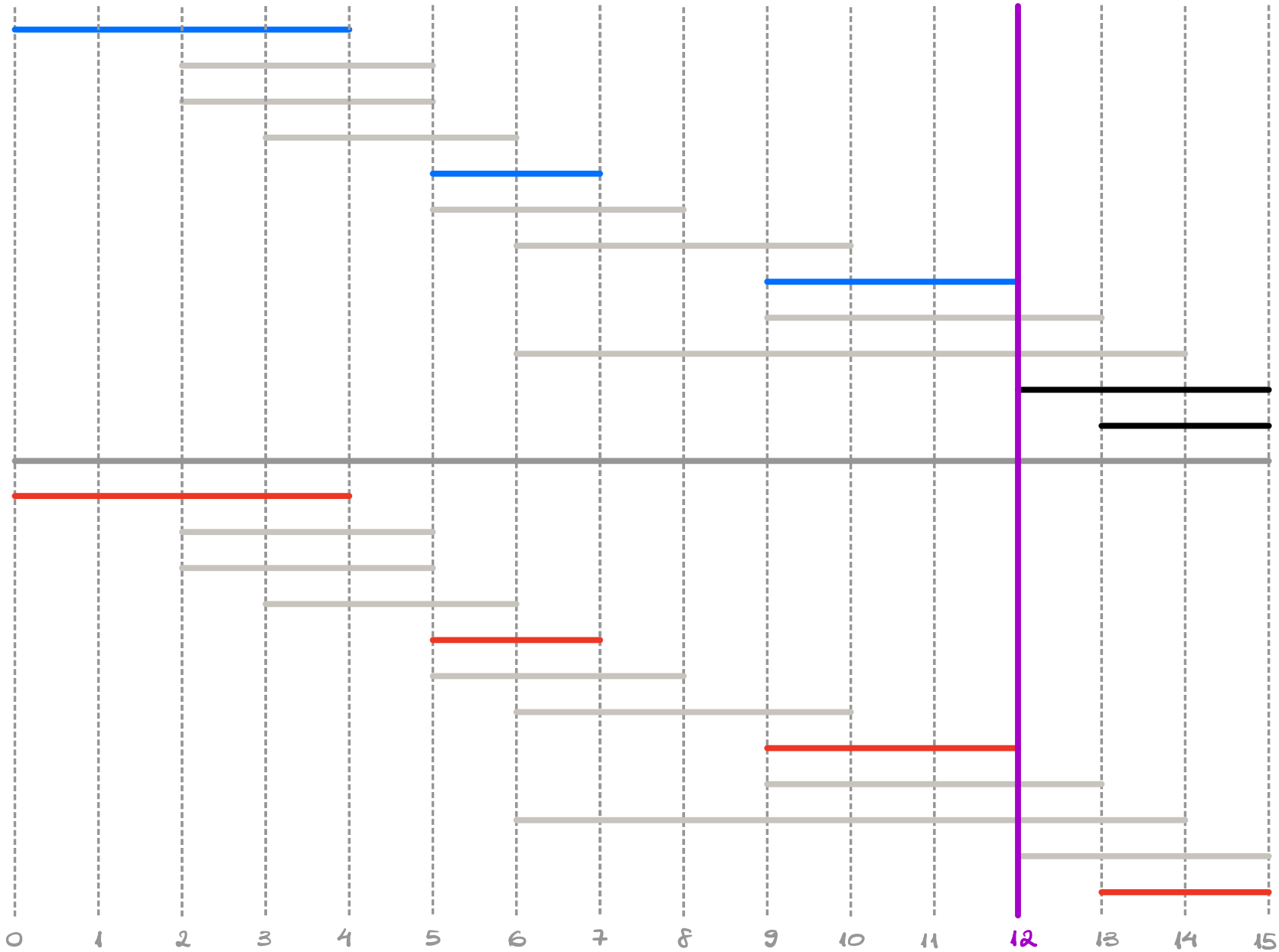
Escolha gulosa e concordância de soluções



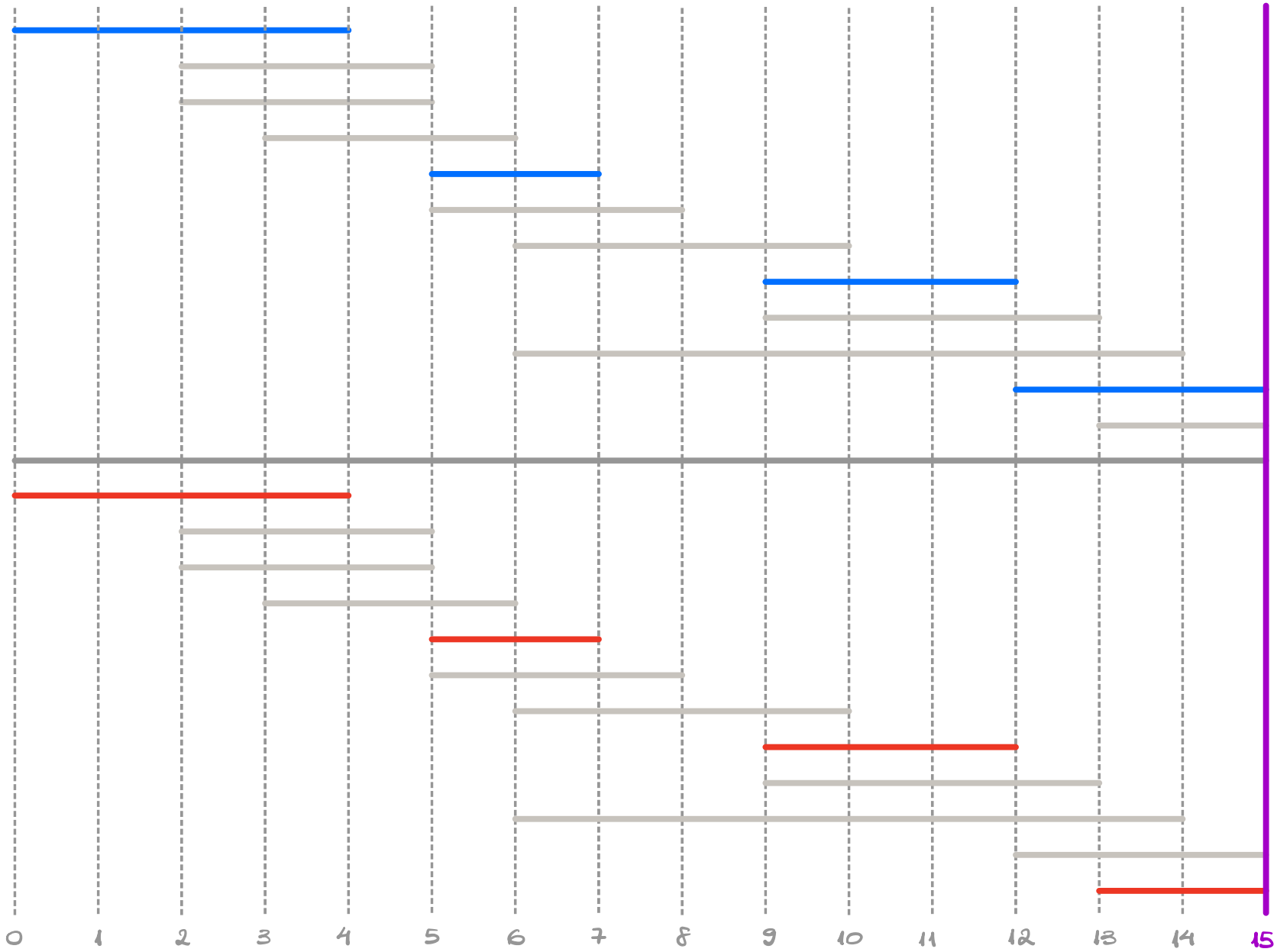
Escolha gulosa e concordância de soluções



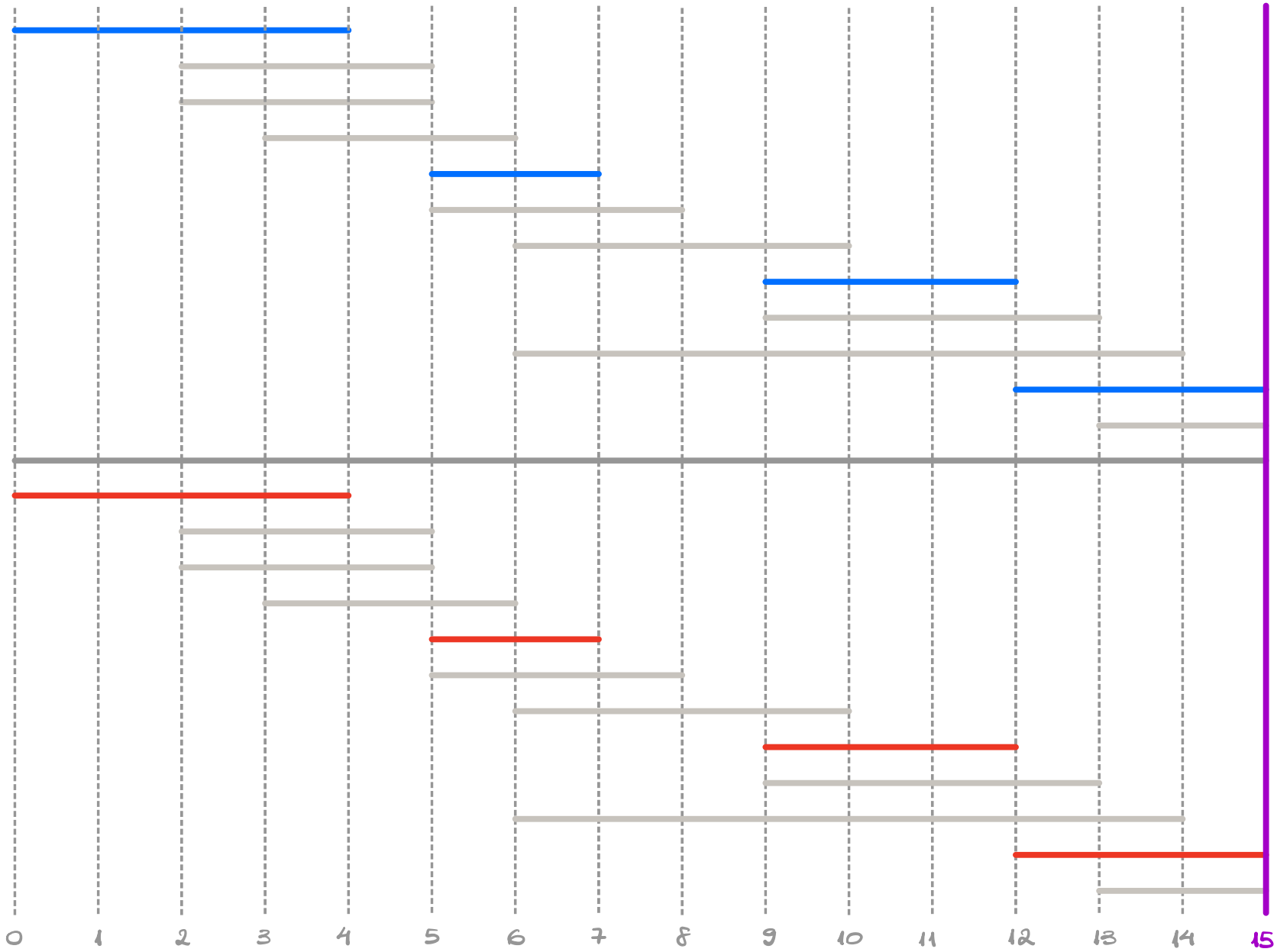
Escolha gulosa e concordância de soluções



Escolha gulosa e concordância de soluções



Escolha gulosa e concordância de soluções



Exercício

1. Mostre um exemplo para os demais critérios gulosos apresentados que prove que o algoritmo obtido usando estes critérios pode produzir um conjunto A que não é máximo.

Coloração de intervalos

Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$ e um inteiro positivo k, uma k-coloração dos intervalos é uma partição deles em k coleções de intervalos disjuntos.

Coloração de intervalos

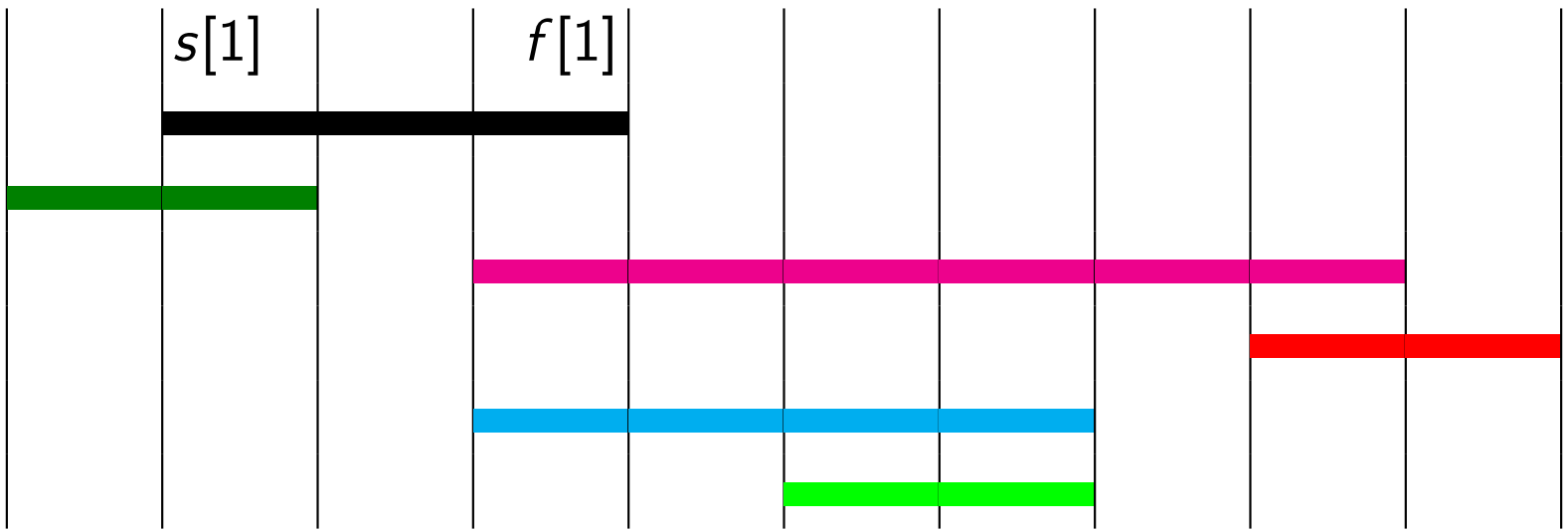
Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$ e um inteiro positivo k, uma k-coloração dos intervalos é uma partição deles em k coleções de intervalos disjuntos.

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar uma k-coloração dos intervalos com k o menor possível.

Coloração de intervalos

Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$ e um inteiro positivo k, uma k-coloração dos intervalos é uma partição deles em k coleções de intervalos disjuntos.

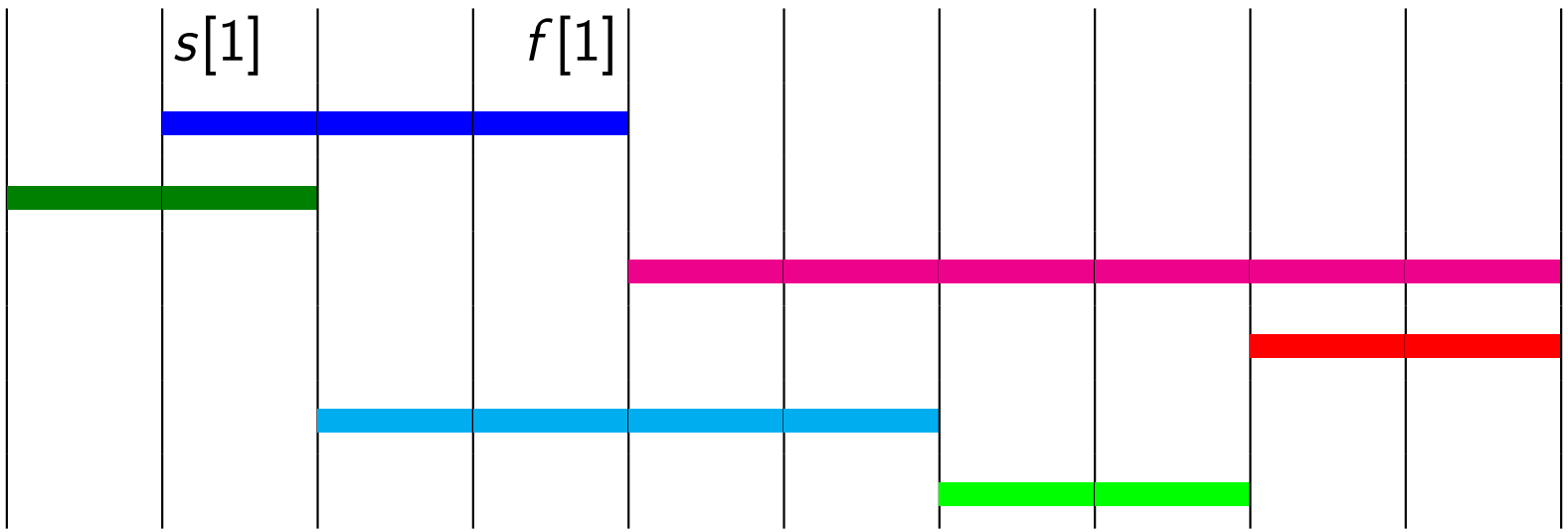
Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar uma k-coloração dos intervalos com k o menor possível.



Coloração de intervalos

Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$ e um inteiro positivo k, uma k-coloração dos intervalos é uma partição deles em k coleções de intervalos disjuntos.

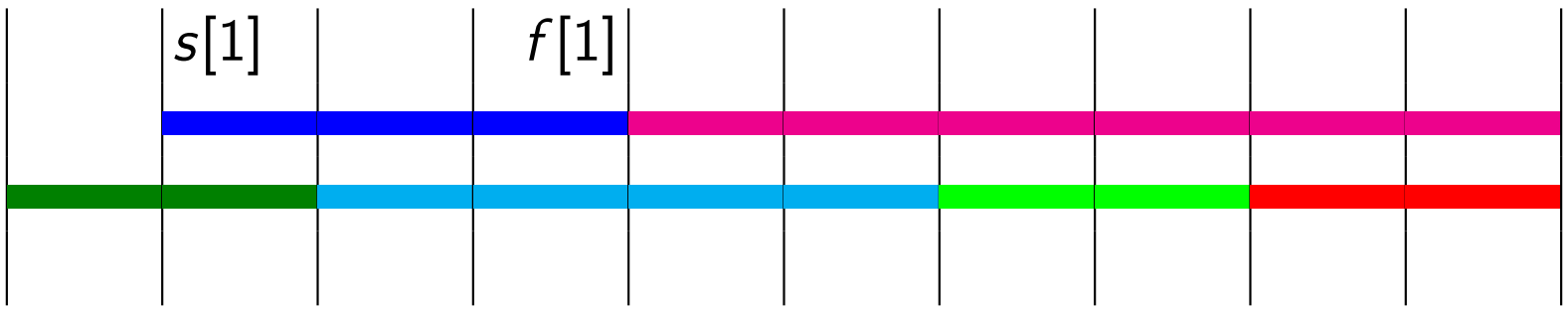
Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar uma k-coloração dos intervalos com k o menor possível.



Coloração de intervalos

Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$ e um inteiro positivo k, uma k-coloração dos intervalos é uma partição deles em k coleções de intervalos disjuntos.

Problema: Dados intervalos $[s[1], f[1]), \ldots, [s[n], f[n])$, encontrar uma k-coloração dos intervalos com k o menor possível.



Solução: 2-coloração.

Coloração de intervalos

Estratégias gulosas:

- Encontre uma coleção disjunta máxima de intervalos, pinte com a próxima cor disponível e repita a idéia para os intervalos restantes.
- Ordene as atividades de maneira que $f[1] \leq f[2] \leq \cdots \leq f[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.
- Ordene as atividades de maneira que $s[1] \leq s[2] \leq \cdots \leq s[n]$ e pinte uma a uma nesta ordem sempre usando a menor cor possível para aquela atividade.

Quais destas estratégias funcionam?

Quais não funcionam?

Algoritmo guloso recursivo

Variáveis globais: m e $c[1\,..\,n]$.

```
GULOSO-R (S)    ▷ S ⊆ {1, ..., n}
1  se S = ∅
2     então m ← 0
3     senão seja i ∈ S com s[i] máximo
4           GULOSO-R(S \ {i})
5           D ← {k ∈ S \ {i} : f[k] > s[i]}
6           se |D| + 1 > m
7              então m ← |D| + 1
8           c[i] ← min({1, ..., m} \ {c[k] : k ∈ D})
```

Chamada: GULOSO-R$(\{1,\ldots,n\})$; a coloração é c.
D = intervalos de $S \setminus \{i\}$ que contêm o ponto $s[i]$.
Consumo de tempo: $T(n) = T(n-1) + O(n)$, ou seja, $O(n^2)$.

Correção: coloração válida

$\omega(S) :=$ máximo, sobre $k \in S$, do número de intervalos de S que contêm $s[k]$.

Invariante. Após GULOSO-R(S): (a) $m = \omega(S)$ e
(b) c restrita a S é coloração com cores em $\{1, \ldots, m\}$.

Prova: Indução em $|S|$; por HI, vale para $S' := S \setminus \{i\}$.

Como $s[p] \leq s[i]$ para $p \in S'$: p contém $s[i] \iff p \in D$.

(a) i só contém $s[k]$ se $s[k] = s[i]$, e $s[i]$ está em $|D| + 1$ intervalos.
Logo $\omega(S) = \max\{\omega(S'), |D| + 1\}$ (linhas 6–7).

(b) D usa $\leq |D| \leq m - 1$ cores: há cor livre na linha 8.
$c[p] = c[i]$, $p \in S' \Rightarrow p \notin D \Rightarrow f[p] \leq s[i]$: disjuntos. □

Correção: número mínimo de cores

Ao final, $m = \omega(\{1, \ldots, n\})$: existe k tal que $s[k]$ pertence a m intervalos.

Esses m intervalos se intersectam dois a dois (todos contêm $s[k]$), logo recebem cores distintas em qualquer coloração.

Portanto toda coloração usa pelo menos m cores.

GULOSO-R usa no máximo m cores, logo é ótimo.

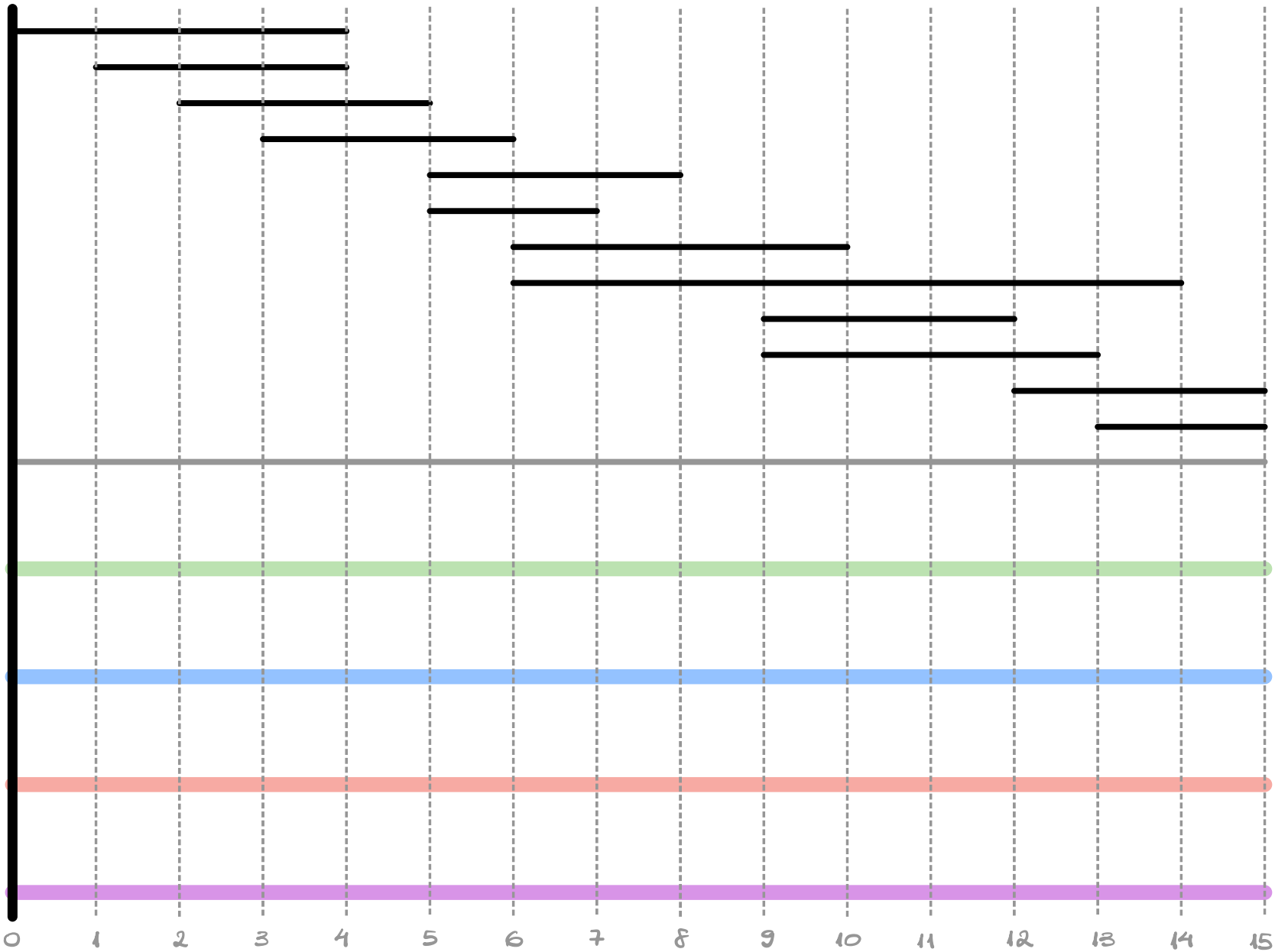
Observação. Desenrolando a recursão, os intervalos são coloridos em ordem crescente de s, sempre com a menor cor livre em $s[i]$.

Isso é exatamente o que GULOSO faz a seguir: a cor j está livre em $s[k] \iff \ell[j] \leq s[k]$.

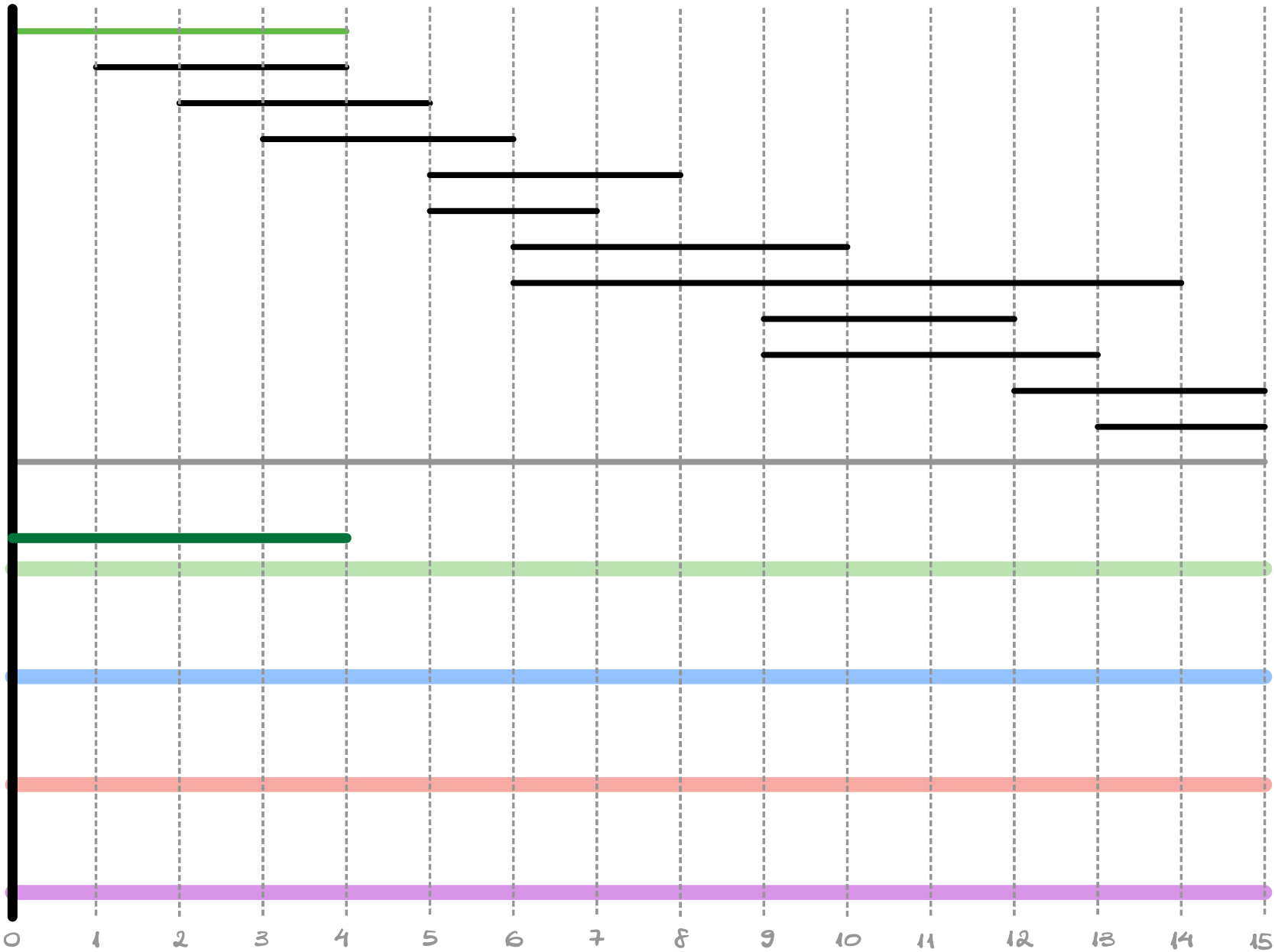
Algoritmo guloso

```
GULOSO (s, f, n)
 1  ORDENE(s, f, n)      ▷ s[1] ≤ s[2] ≤ ··· ≤ s[n]
 2  m ← 0
 3  ℓ[1] ← 0
 4  para k ← 1 até n faça
 5      j ← 1
 6      enquanto ℓ[j] > s[k] faça
 7          j ← j + 1
 8      se j > m
 9          então m ← j
10                  ℓ[m + 1] ← 0
11      ℓ[j] ← f[k]
12      c[k] ← j
13  devolva c
```

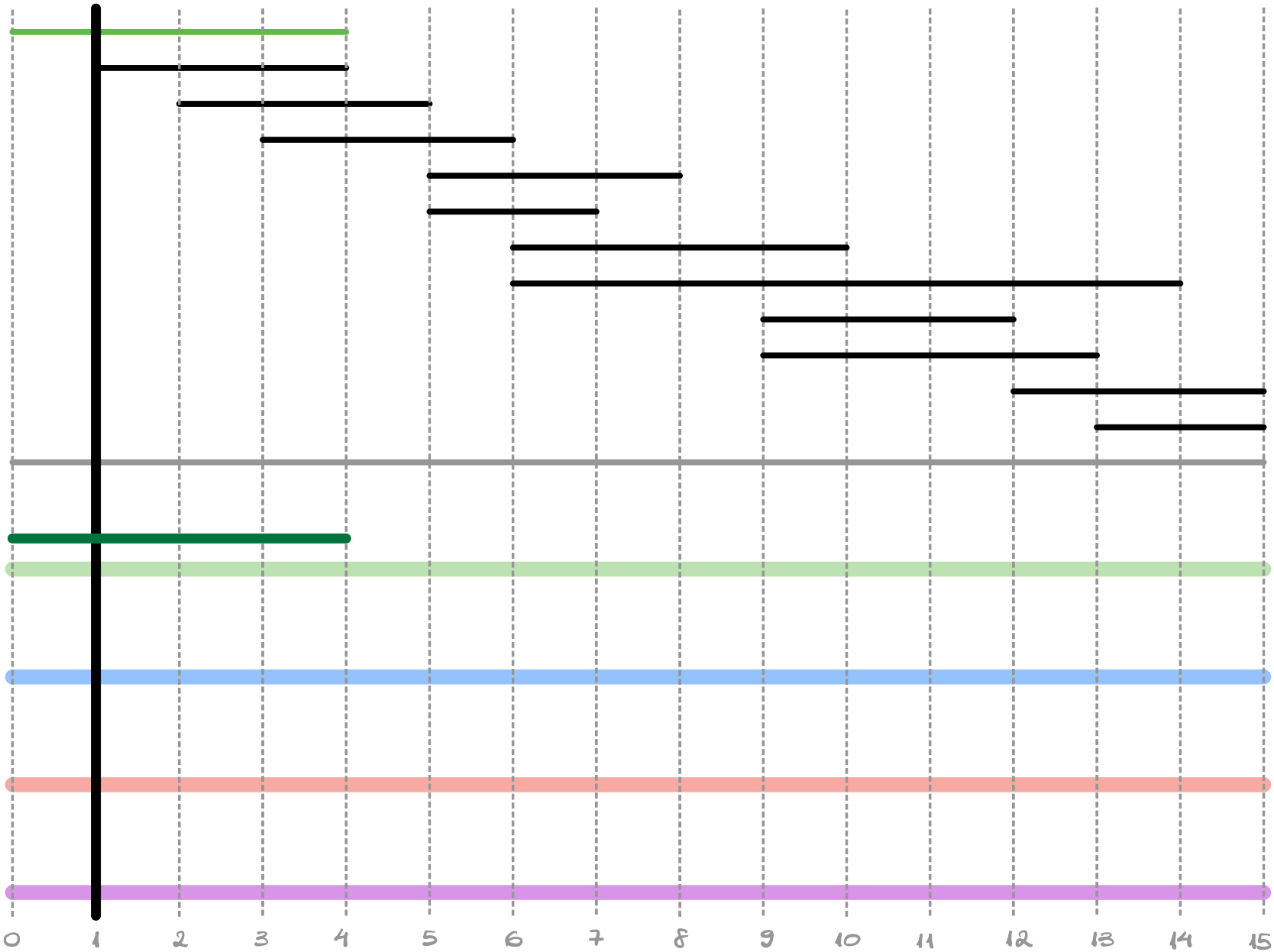
Escolha gulosa



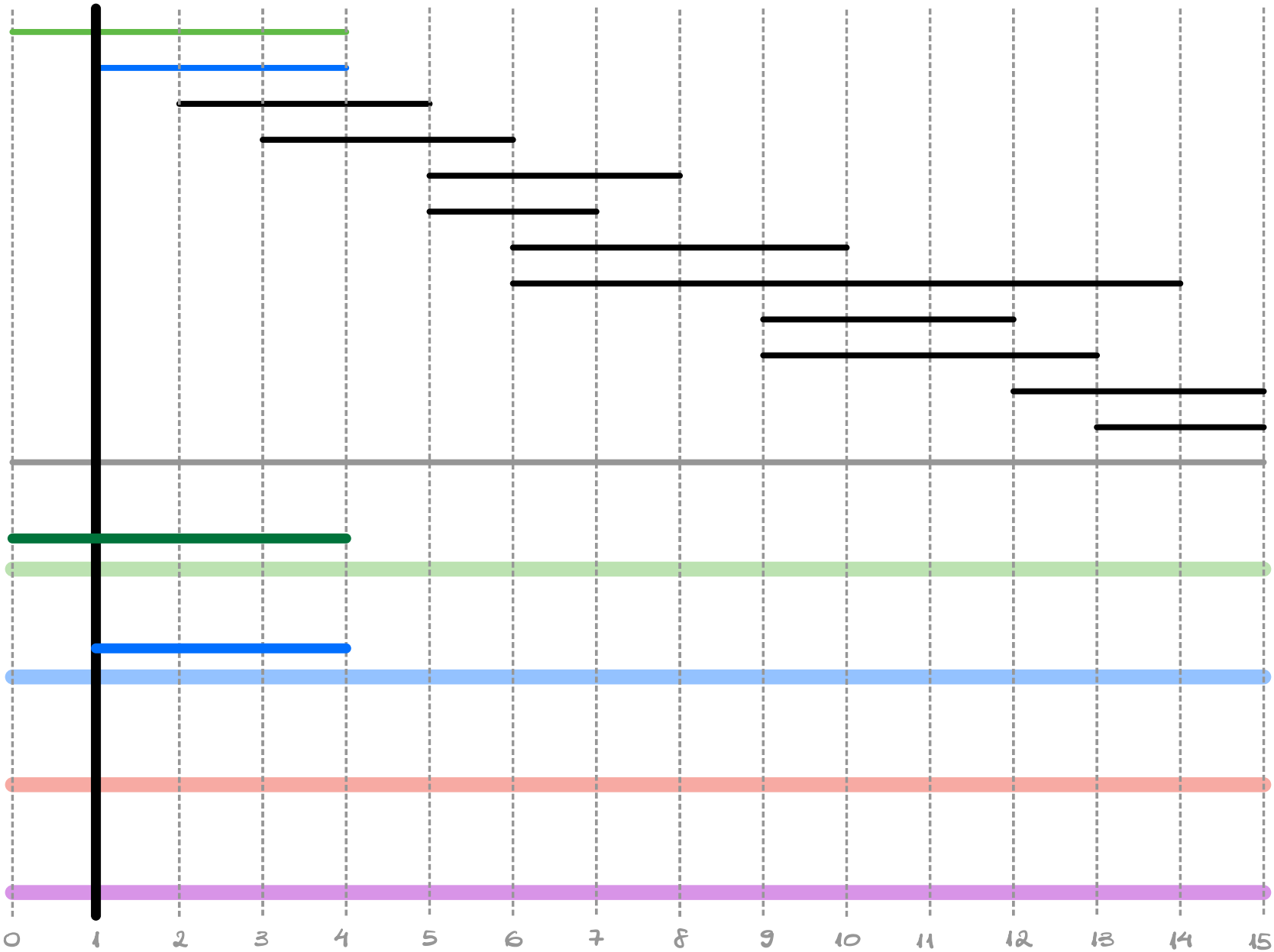
Escolha gulosa



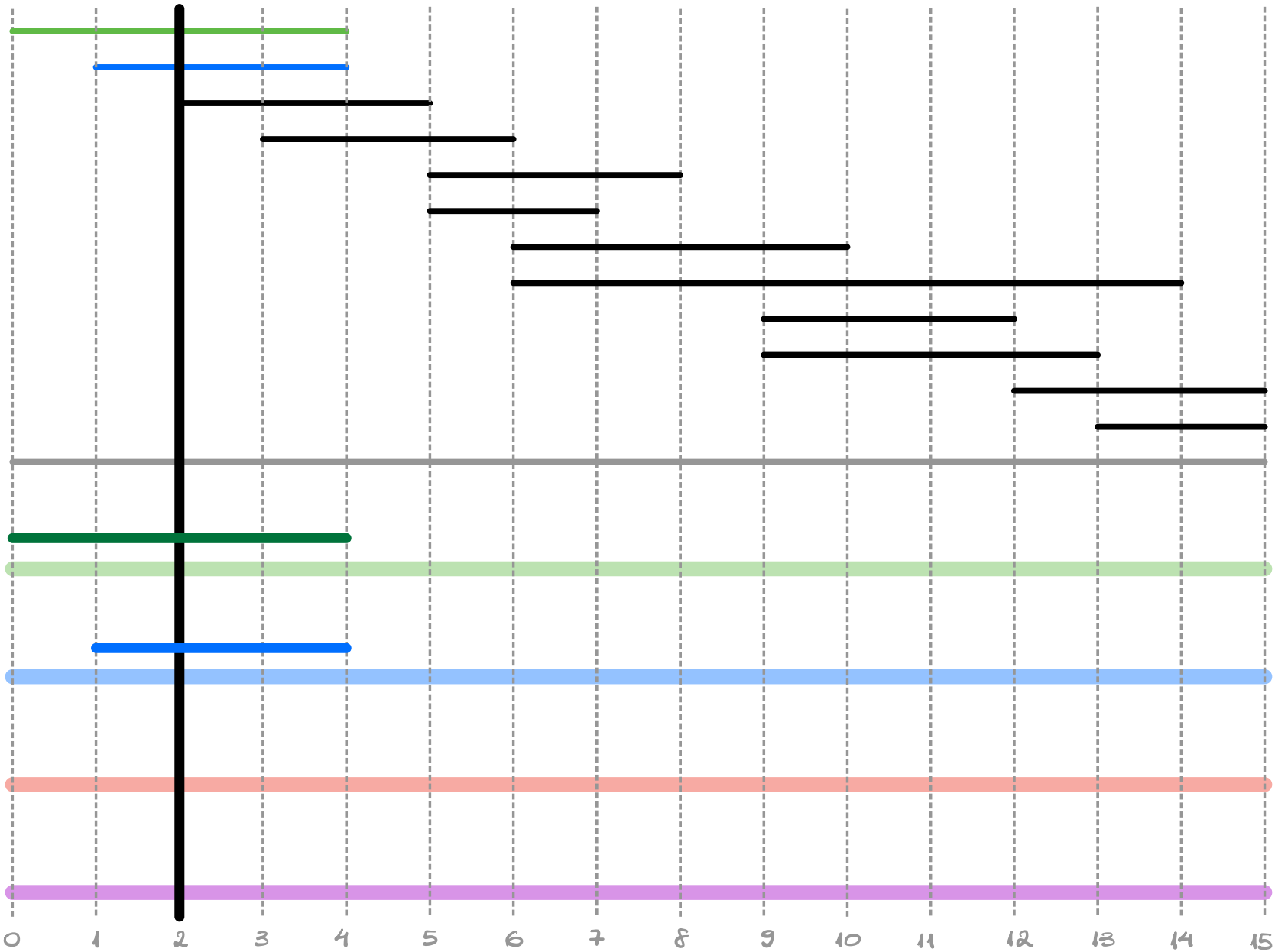
Escolha gulosa



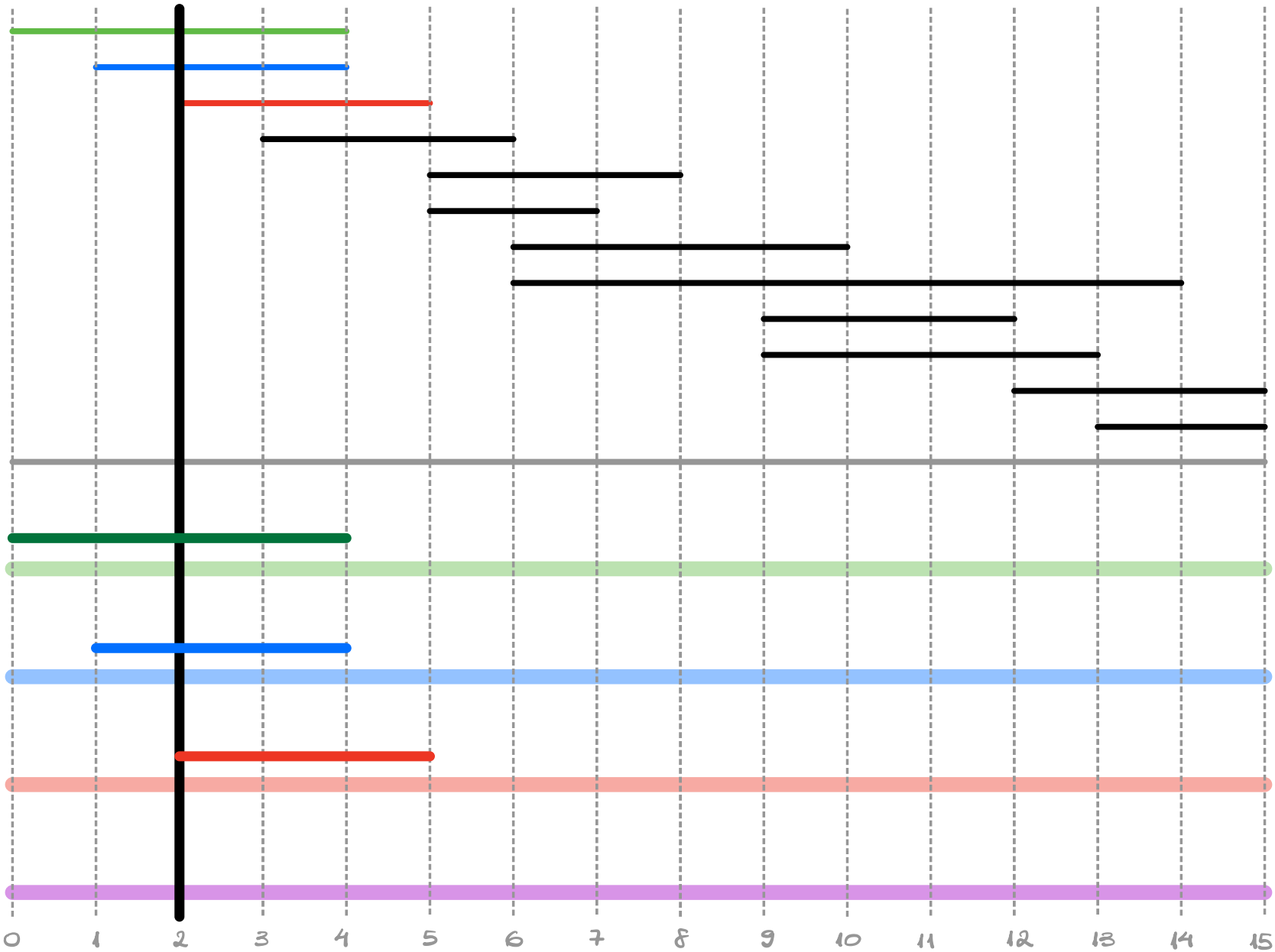
Escolha gulosa



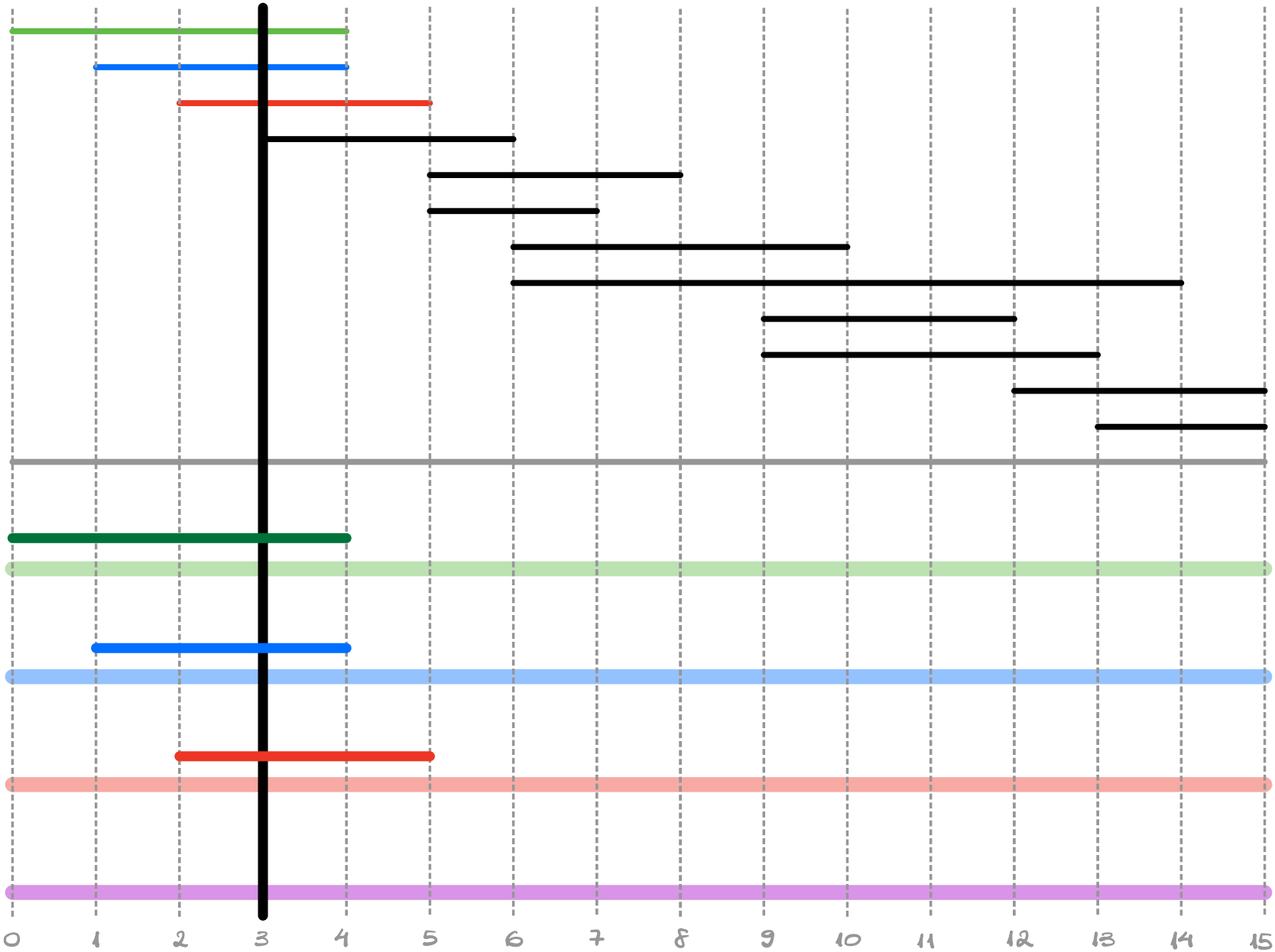
Escolha gulosa



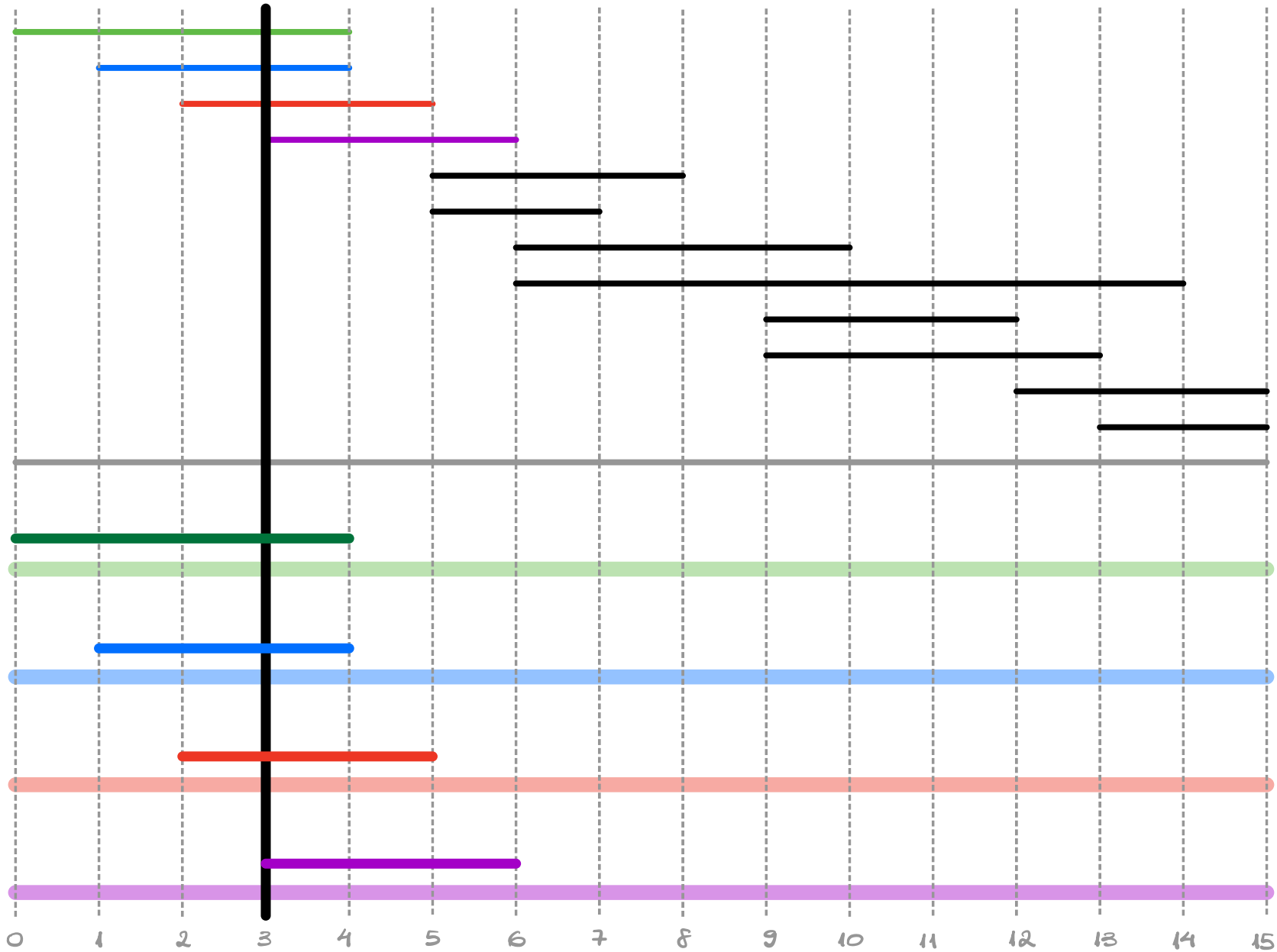
Escolha gulosa



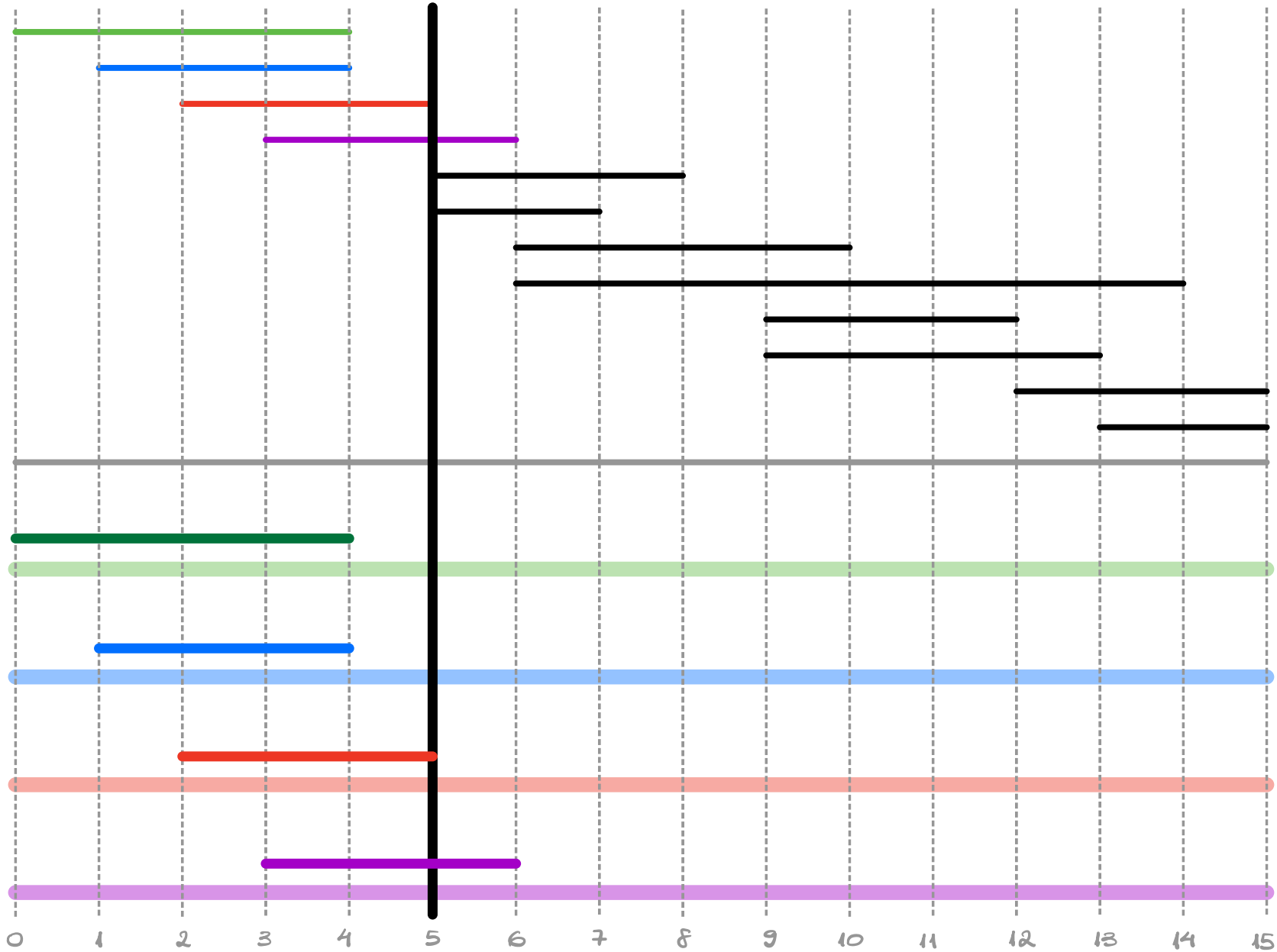
Escolha gulosa



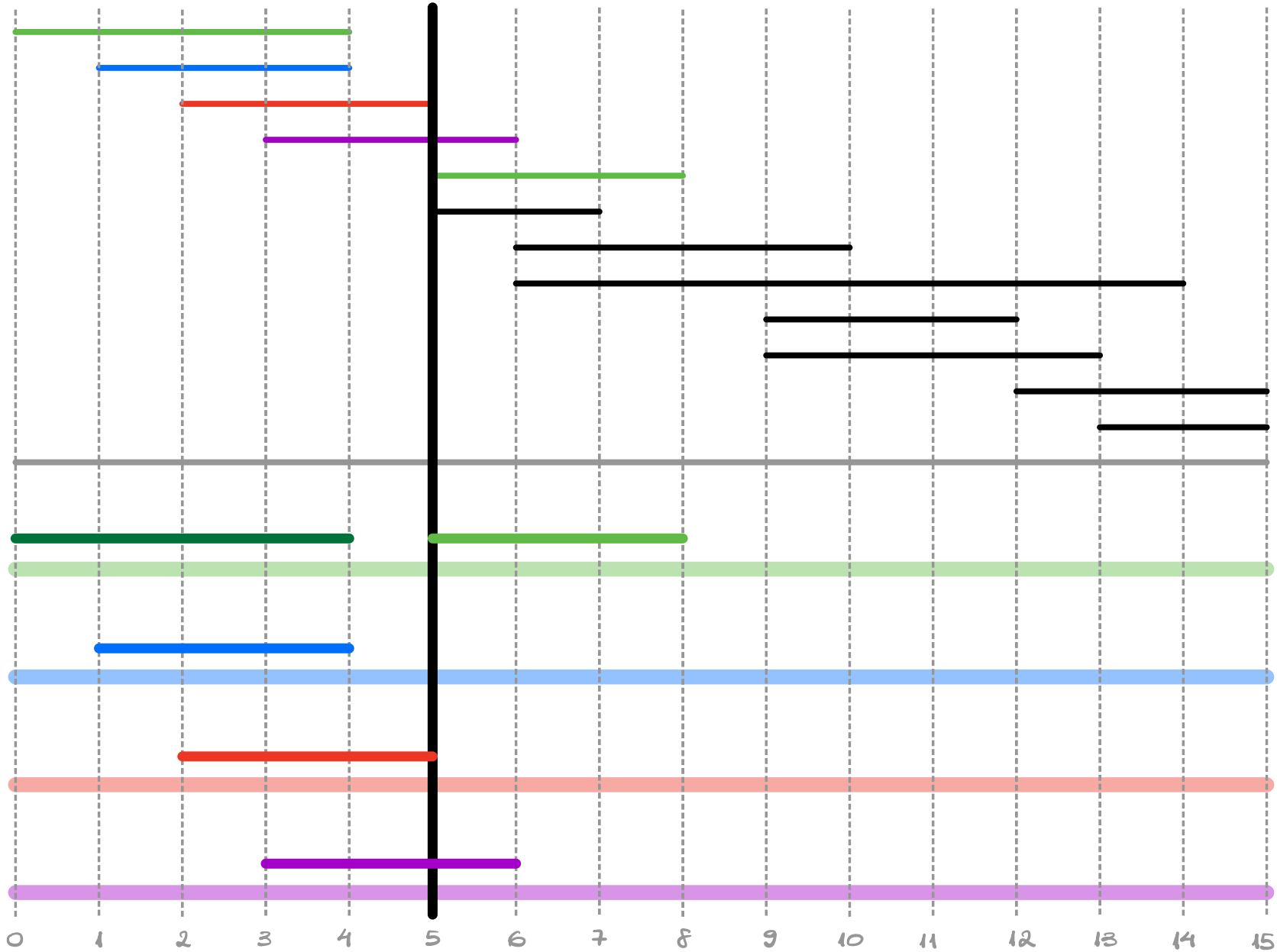
Escolha gulosa



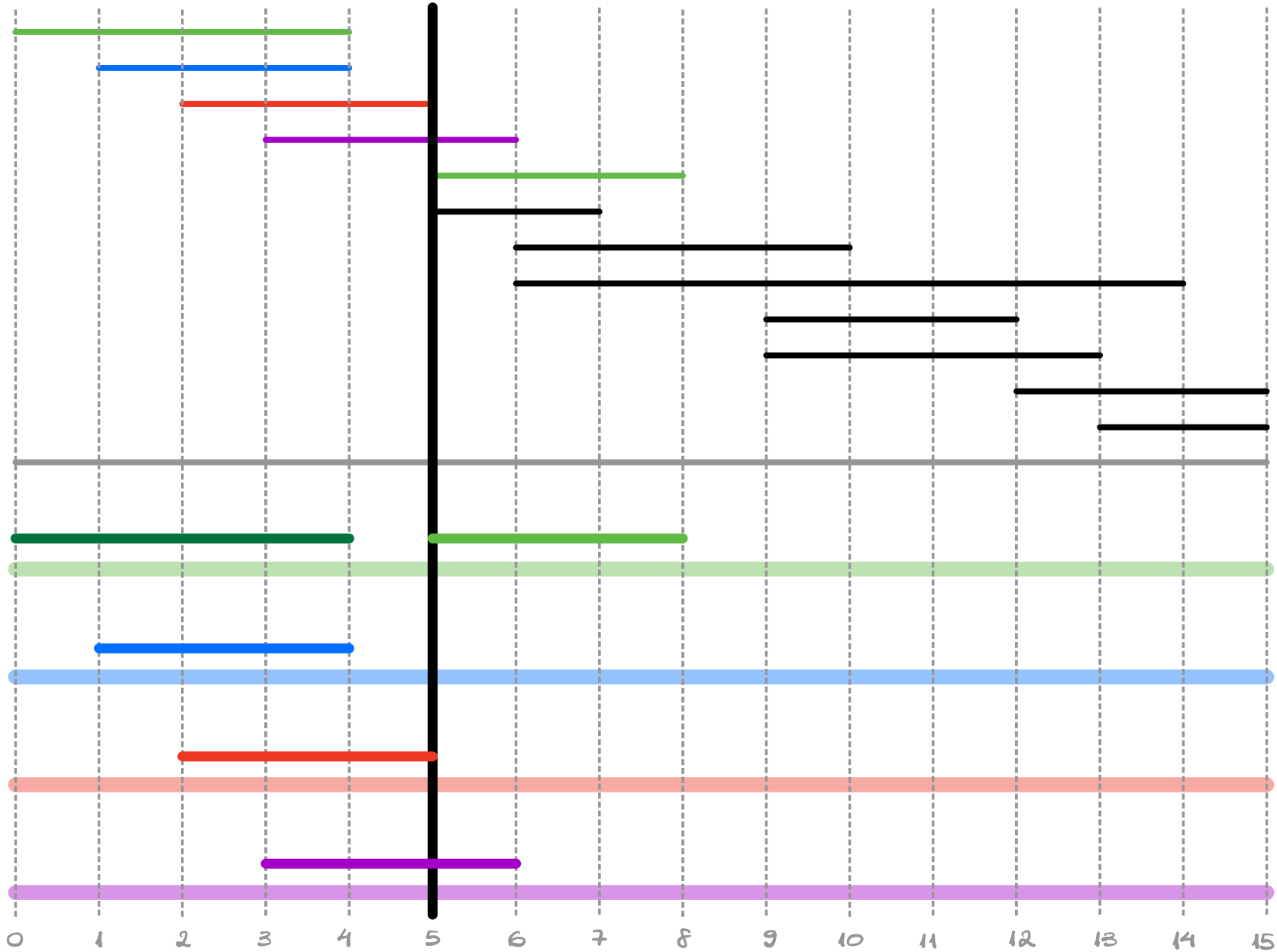
Escolha gulosa



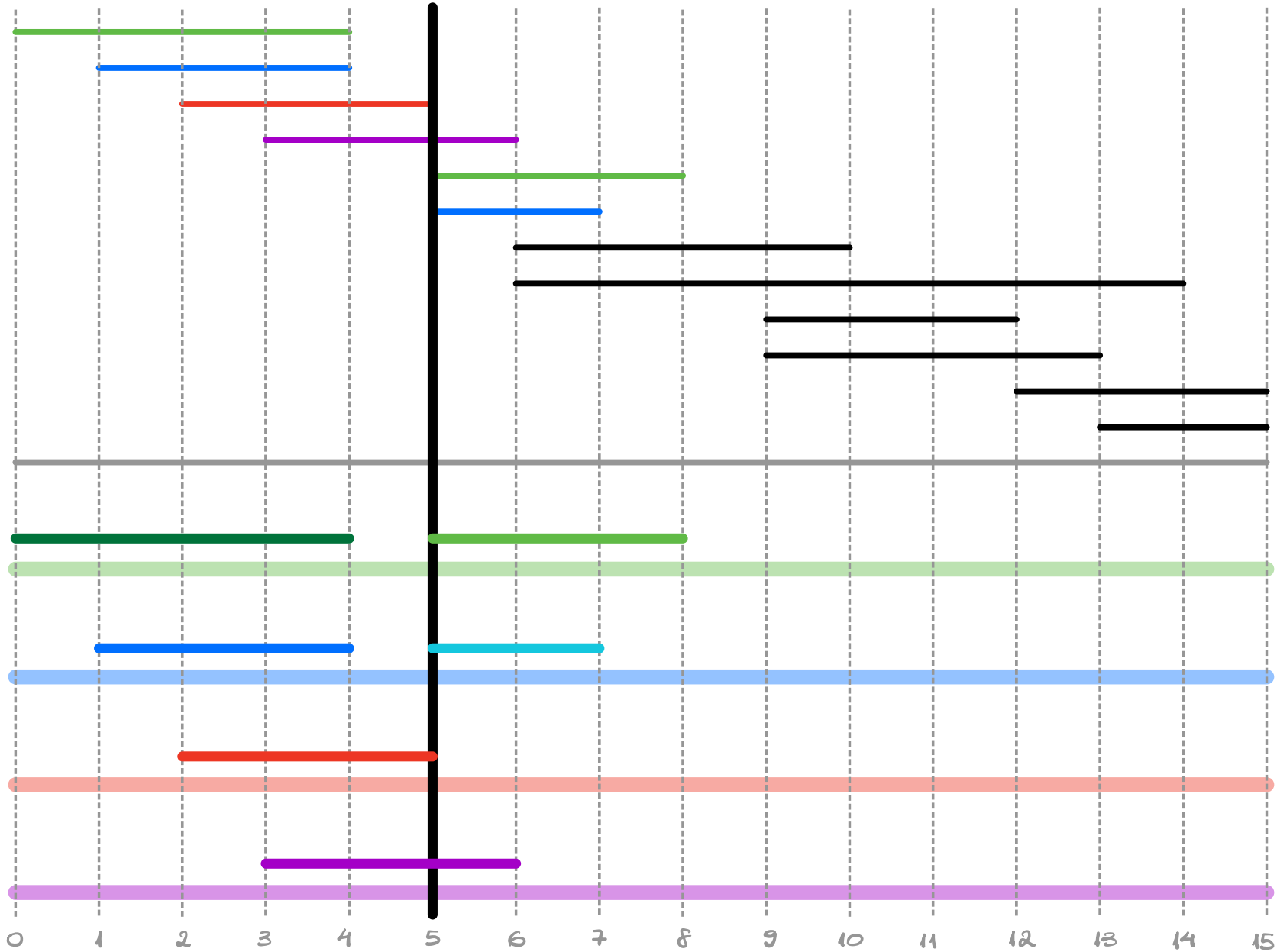
Escolha gulosa



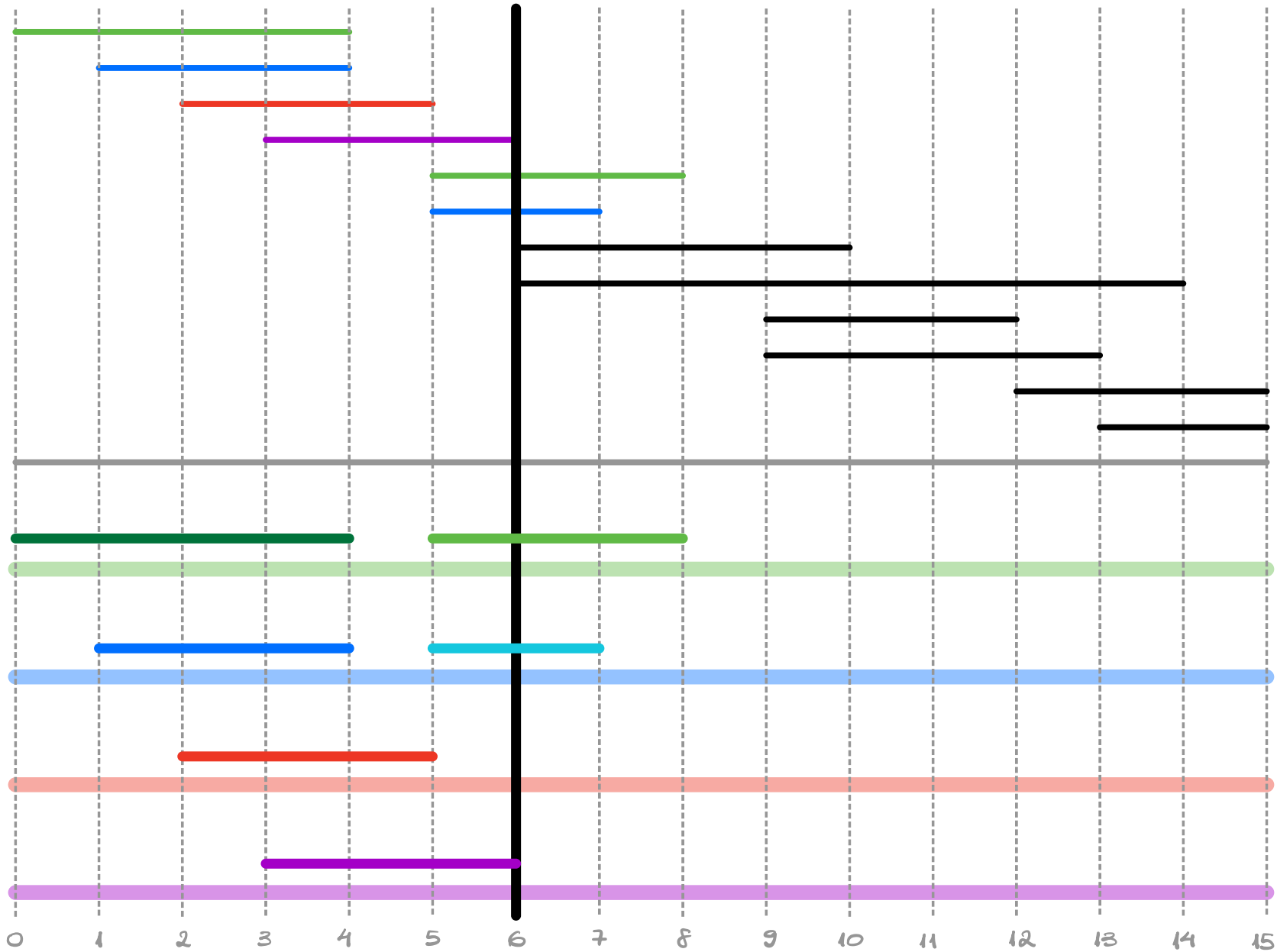
Escolha gulosa



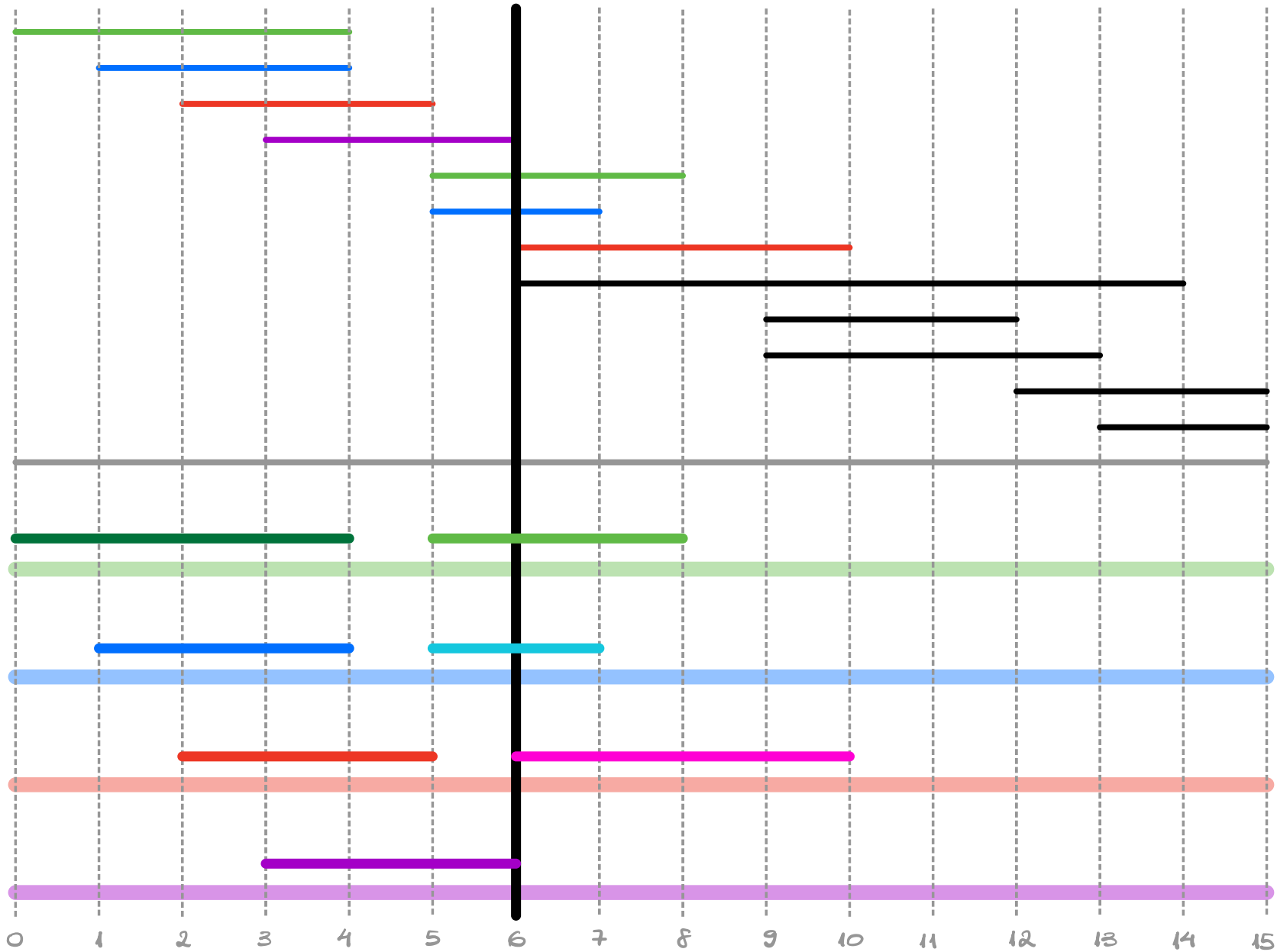
Escolha gulosa



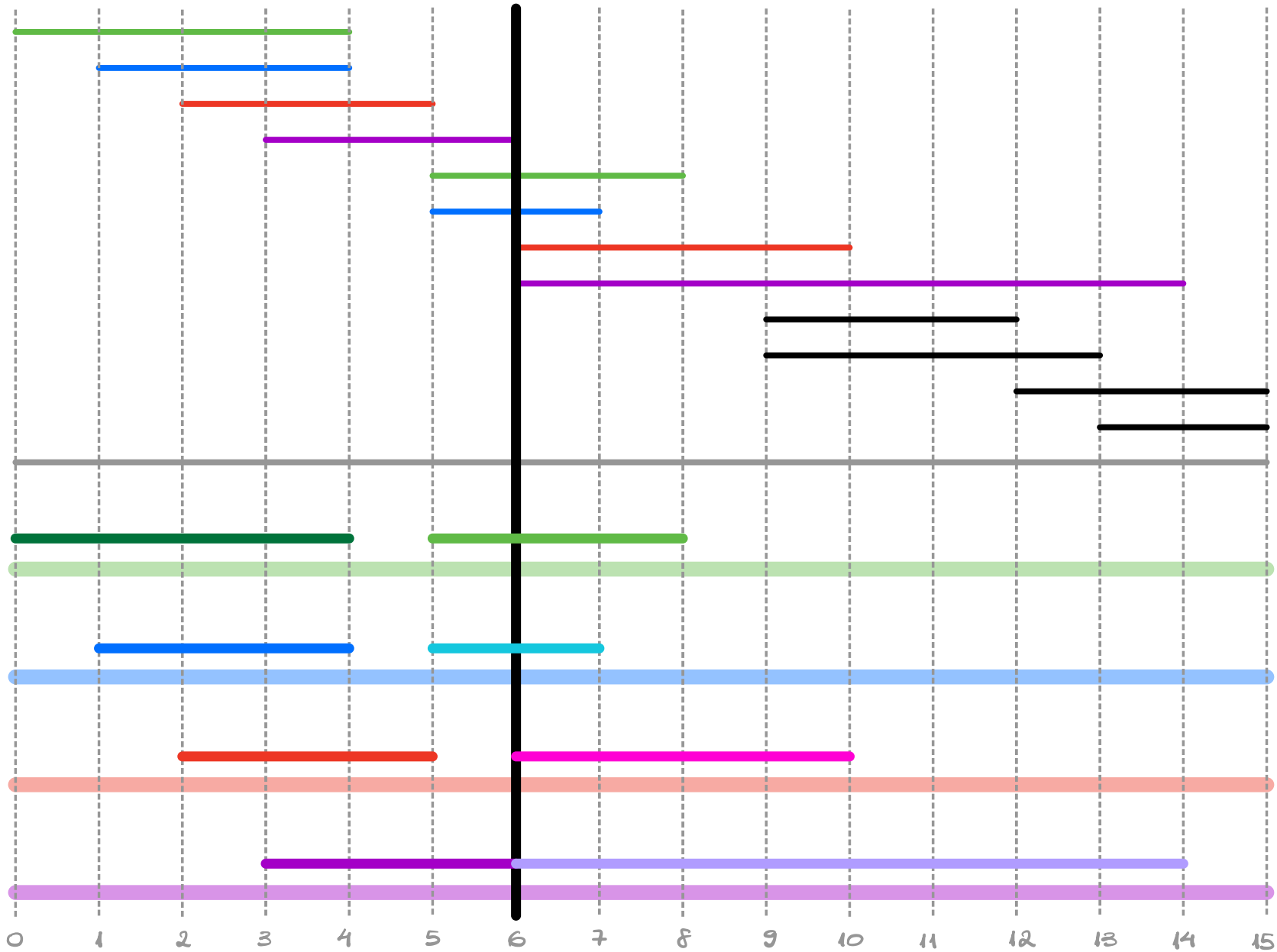
Escolha gulosa



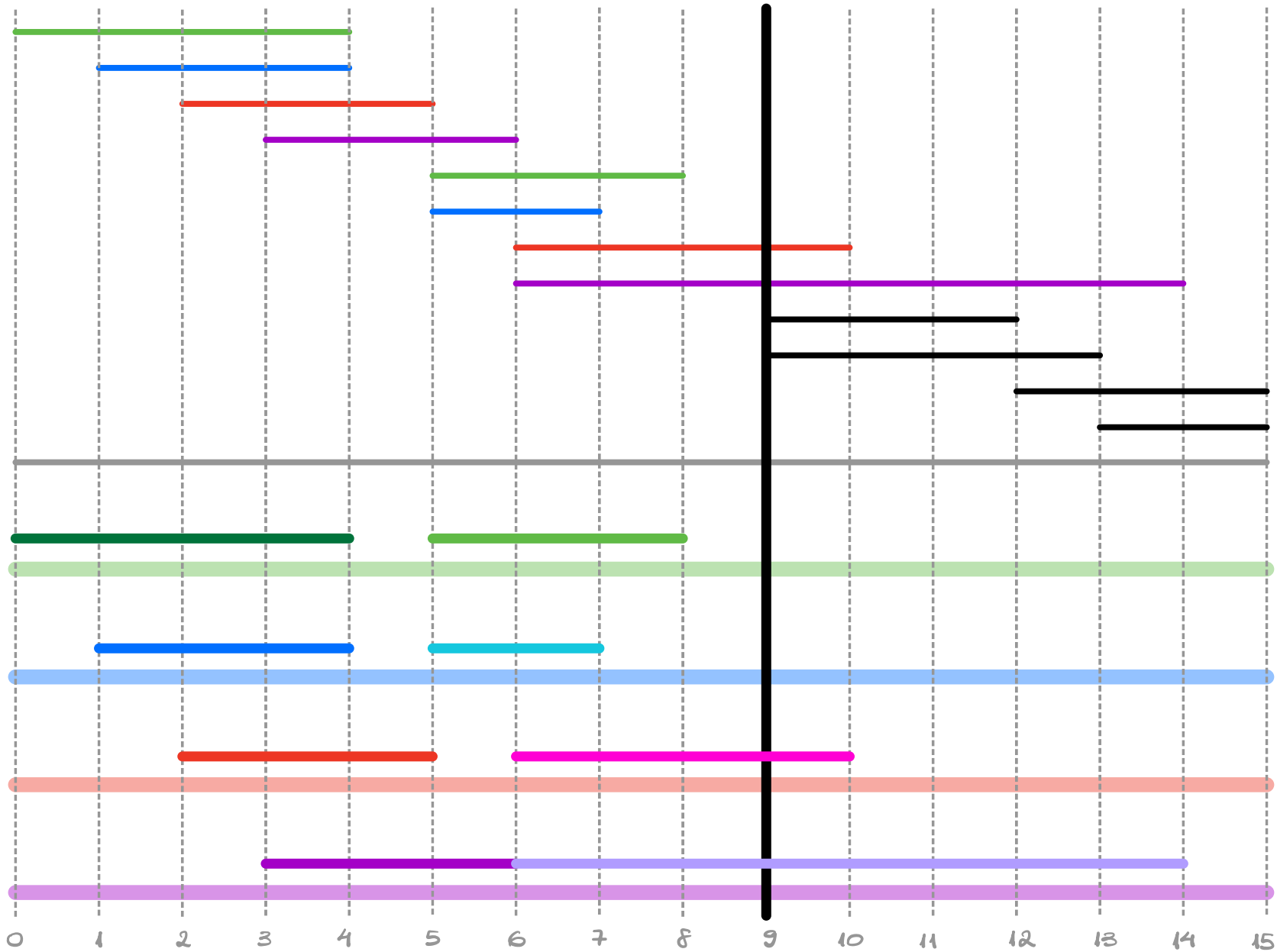
Escolha gulosa



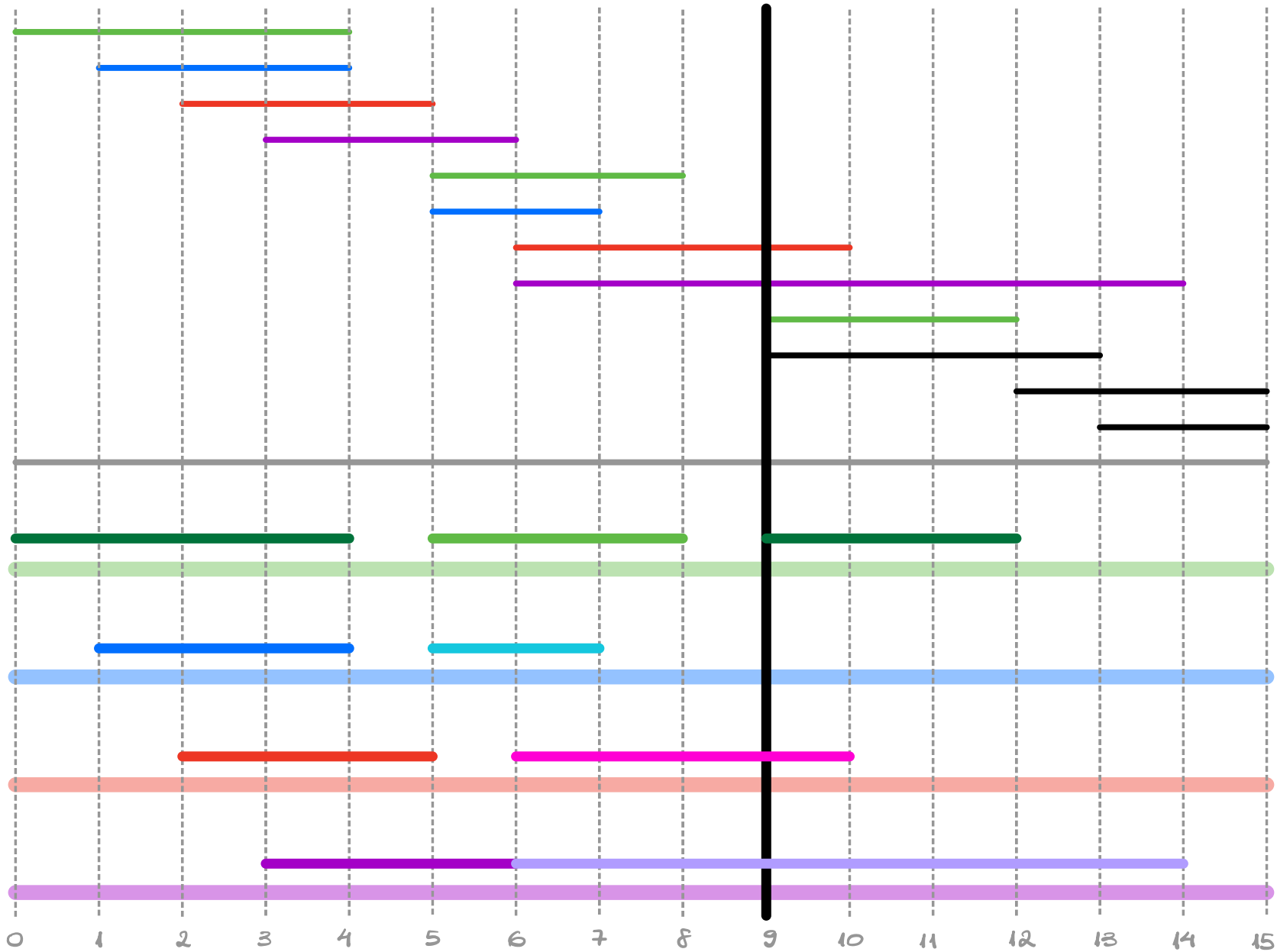
Escolha gulosa



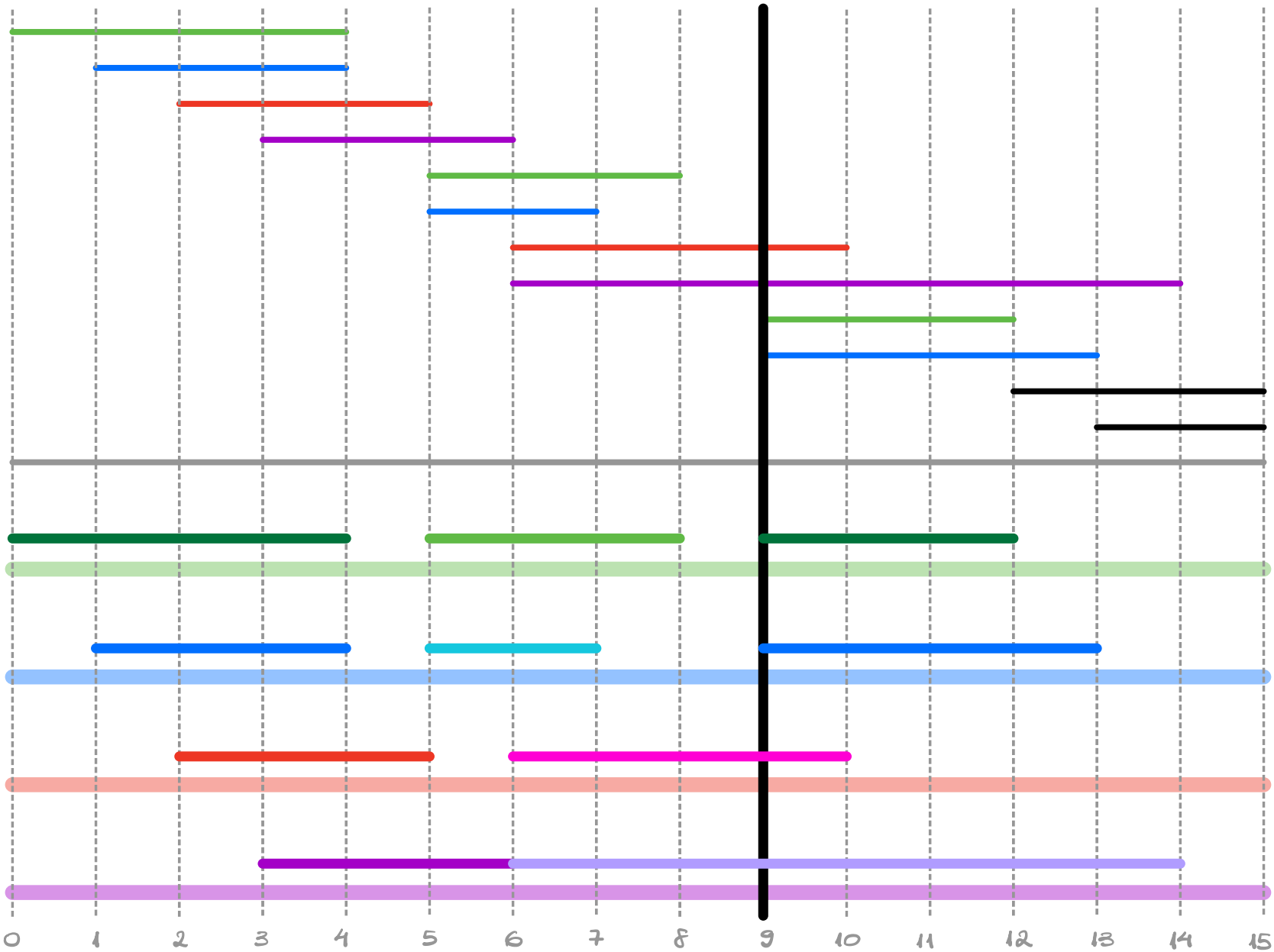
Escolha gulosa



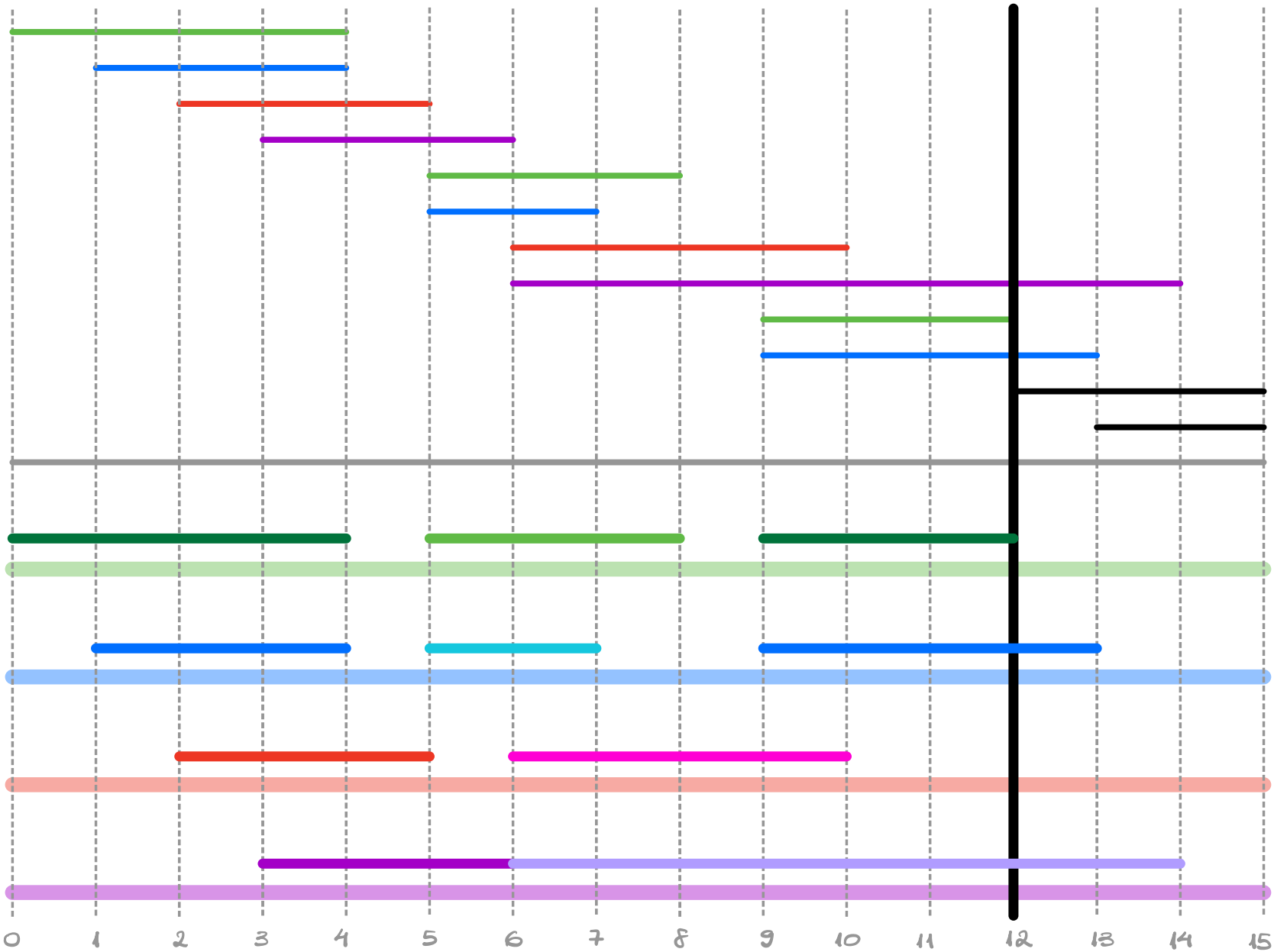
Escolha gulosa



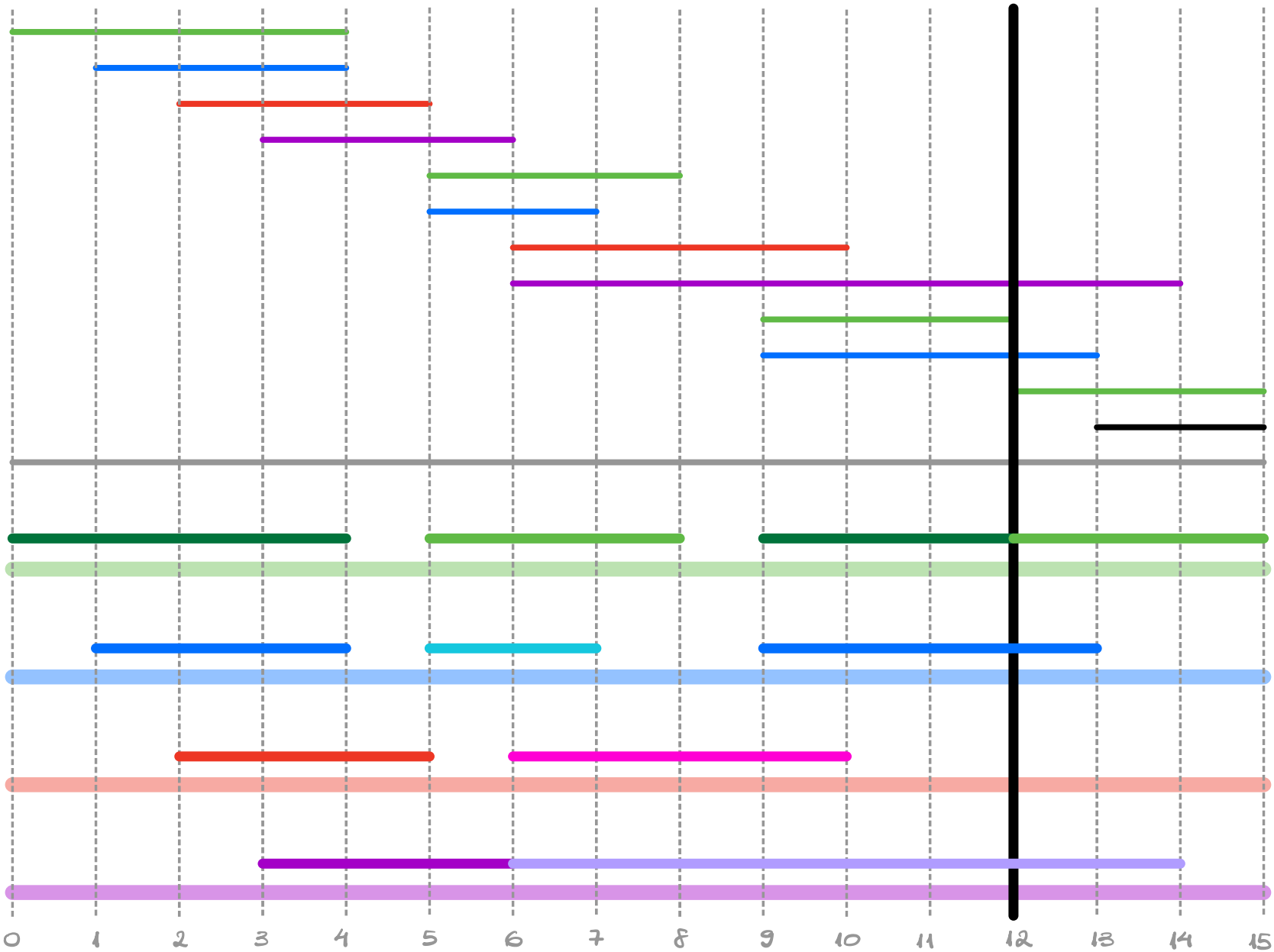
Escolha gulosa



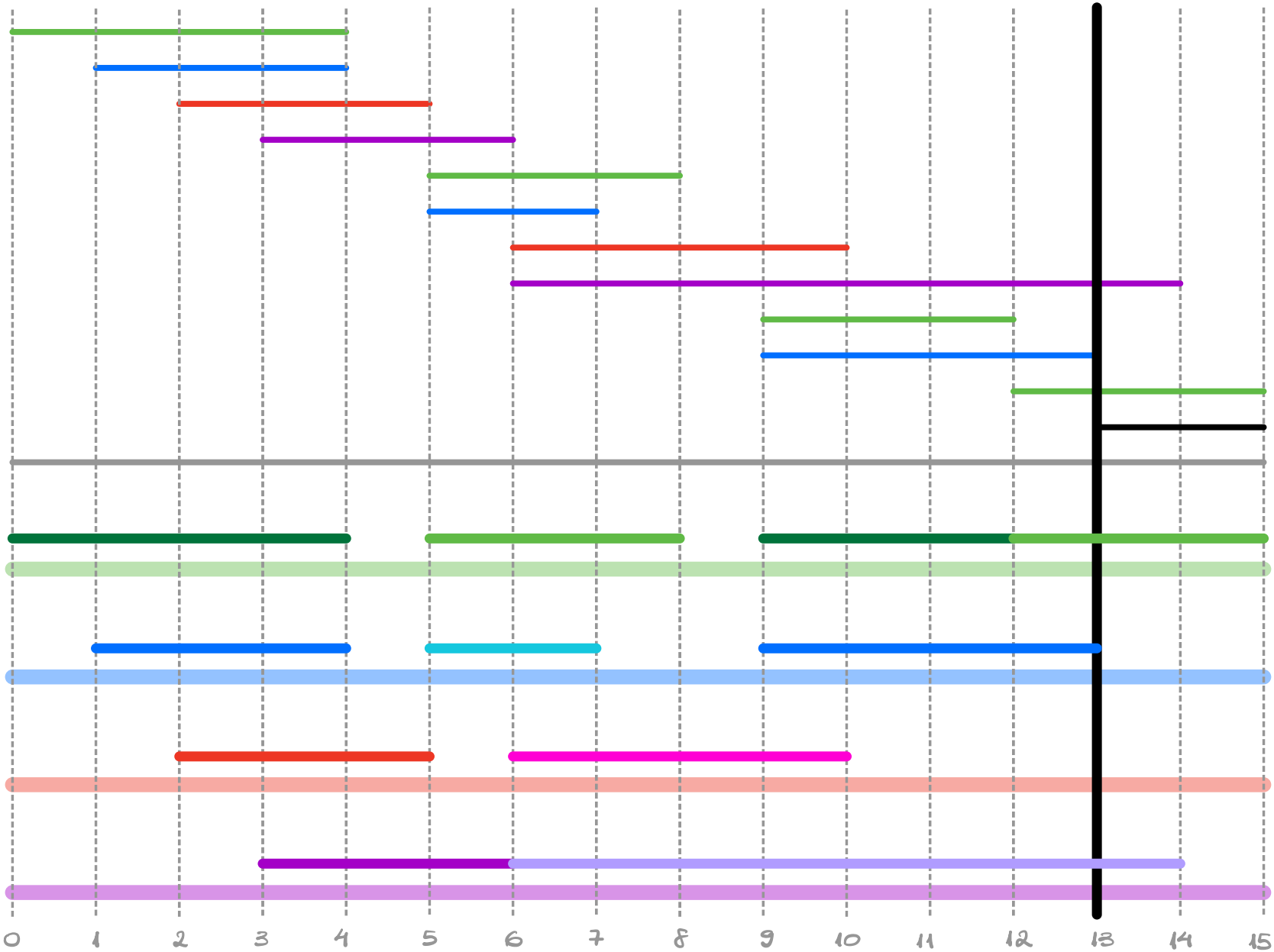
Escolha gulosa



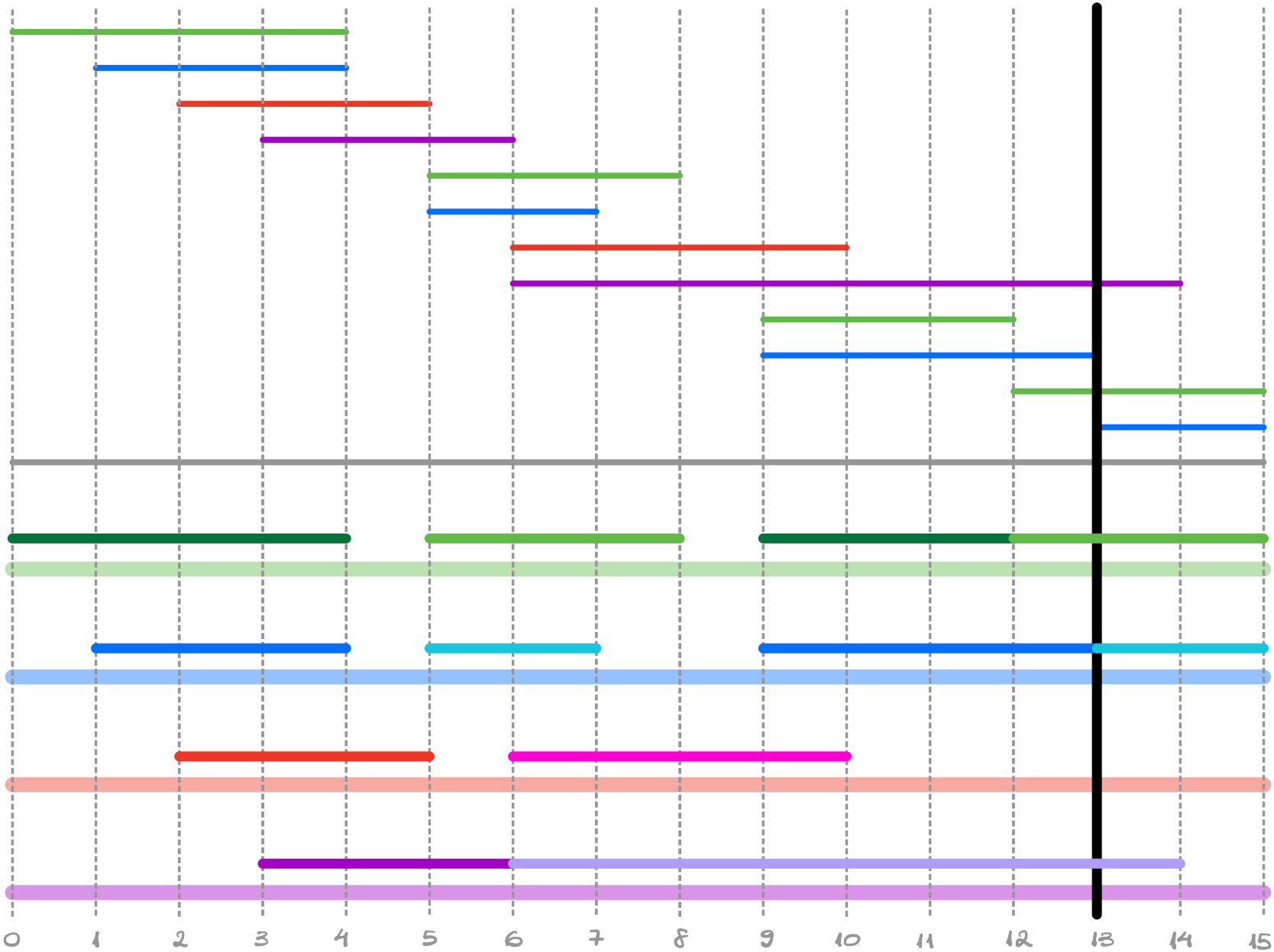
Escolha gulosa



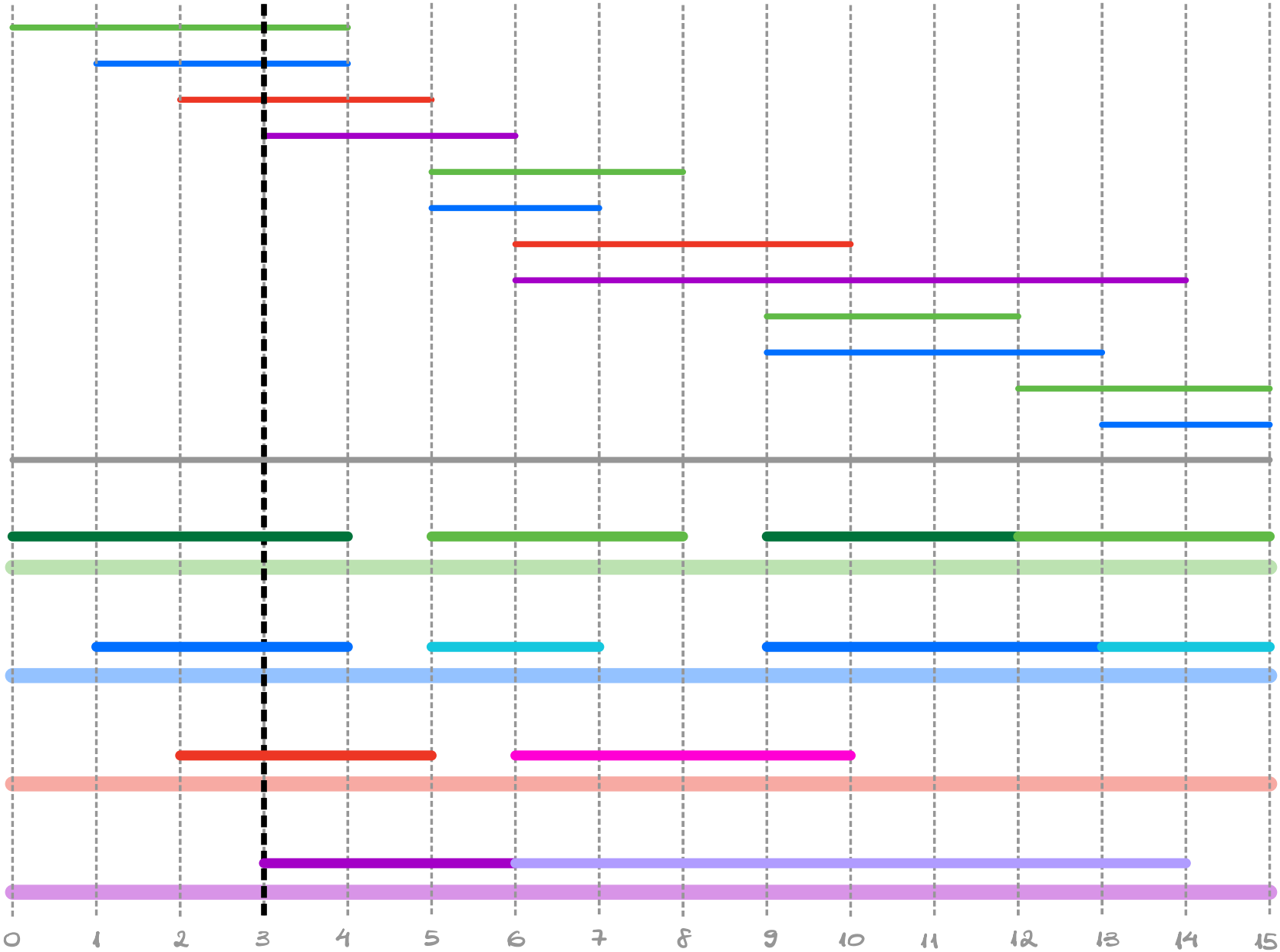
Escolha gulosa



Escolha gulosa



Escolha gulosa



Exercícios

1. Considere o algoritmo do slide anterior. Modifique-o para que, além de c, ele devolva
   um conjunto S de m intervalos e um instante t tal que
   $s[i] \leq t < f[i]$ para todo i em S.
2. Numa versão anterior dos slides desta aula, o algoritmo do problema da coloração ordenava os intervalos pelo valor de f em vez de pelo valor de s. Isso faz diferença? Ou seja, a versão do algoritmo da página anterior que ordena os intervalos por f em vez de s também dá uma resposta correta? Prove ou dê um contra-exemplo.

Guloso recursivo com heaps

Supõe $s[1] \leq s[2] \leq \cdots \leq s[n]$ (chame ORDENE(s, f, n) antes).
Variáveis globais: m, $c[1\,..\,n]$ e
H: min-heap dos intervalos ativos, com chave f;
L: min-heap das cores livres.

```
GULOSO-RH (n)
1   se n = 0
2       então m ← 0;  H ← ∅;  L ← ∅
3       senão GULOSO-RH(n − 1)
4             enquanto H ≠ ∅ e f[MIN(H)] ≤ s[n] faça
5                 p ← EXTRAI-MIN(H);  INSERE(L, c[p])
6             se L = ∅
7                 então m ← m + 1;  c[n] ← m
8                 senão c[n] ← EXTRAI-MIN(L)
9             INSERE(H, n)
```

Guloso recursivo com heaps: correção

A chamada recursiva vem antes: os intervalos são coloridos em ordem crescente de s.

Se $f[p] \leq s[n]$, então p não contém $s[k]$ para nenhum $k \geq n$:
p pode sair de H de vez.

Após as linhas 4–5: $H = D = \{k < n : f[k] > s[n]\}$.

Os intervalos de D contêm $s[n]$, logo têm cores distintas:
$L = \{1, \ldots, m\} \setminus \{c[k] : k \in D\}$ e $|L| = m - |D|$.

$L = \emptyset \iff |D| + 1 > m$.

Linhas 6–8 fazem o mesmo que as linhas 6–8 de GULOSO-R:
a mesma coloração, com o mesmo invariante.

Guloso recursivo com heaps: consumo de tempo

Cada intervalo entra e sai de H no máximo uma vez.

Cada cor entra em L ao ser liberada e sai ao ser usada.

Total: $O(n)$ operações de heap, cada uma $O(\lg n)$.

ORDENE(s, f, n): $\Theta(n \lg n)$.

O consumo de tempo de
ORDENE(s, f, n) seguido de GULOSO-RH(n) é $\Theta(n \lg n)$.

Observação. Com a recursão desenrolada, isto é o laço "**para** $k \leftarrow 1$ **até** n" de GULOSO.