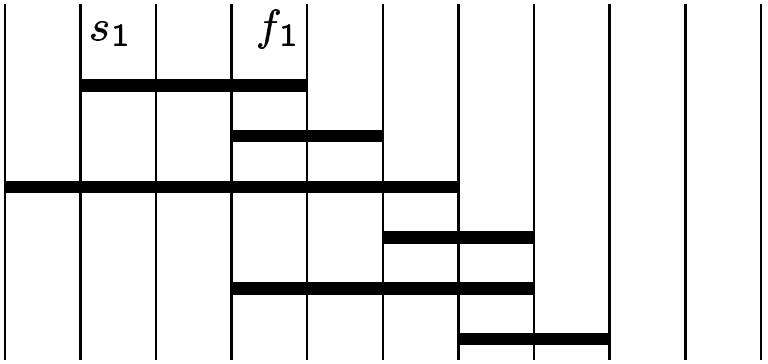
Algoritmo guloso, EXEMPLO 2:

# Problema dos intervalos disjuntos

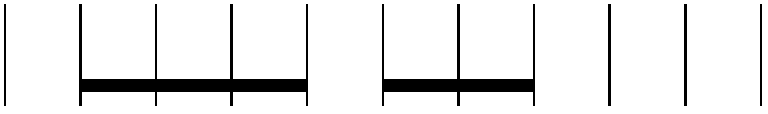
Dados intervalos $[s_1, f_1), \ldots, [s_n, f_n)$,
encontrar uma coleção máxima de intervalos disjuntos dois a dois.

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

Exemplo:




Solução:



Se cada intervalo é uma "atividade", queremos coleção disjunta máxima de atividades compatíveis
($i$ e $j$ são compatíveis se $f_i \leq s_j$)

Análise do problema:

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

Propriedade da escolha gulosa:

> Se $f_i$ é mínimo em $S$ então
> $m$ está em alguma solução ótima.

Propriedade da subestrutura ótima:

> Suponha que $f_i$ é mínimo em $S$.
> Se $A$ é solução ótima de $S$ e $A \ni i$
> então $A - \{i\}$ é solução ótima de $S_i$.

Notação: $S_i := \{k : s_k \geq f_i\}$
(todos os intervalos "à direita" de $i$)

Prove as propriedades!

**Algoritmo guloso:**

Supõe $f_1 \leq \cdots \leq f_n$

Primeira versão:

```
FEIO-INTERVALOS-DISJUNTOS (s, f, n)
1  A ← ∅
2  i ← 1
3  enquanto i ≤ n faça
4      A ← A ∪ {i}
5      m ← i + 1
6      enquanto m ≤ n e s_m < f_i faça
7          m ← m + 1
8      i ← m
9  devolva A
```

Correta, eficiente, mas deselegante.

Segunda versão: mais elegante

Supõe $f_1 \leq \cdots \leq f_n$

```
INTERVALOS-DISJUNTOS (s, f, n)
1  A ← {1}
2  i ← 1
3  para m ← 2 até n faça
4     se s_m ≥ f_i
5        então A ← A ∪ {m}
6              i ← m
7  devolva A
```

Consumo de tempo: $\Theta(n)$

Nome do algoritmo no CLRS: Greedy-Activity-Selector

# TAREFA 19

**Exercício 19.A** [CLRS 16.1-1]
Escreva um algoritmo de programação dinâmica para resolver o problema dos intervalos disjuntos. (Versão simplificadda do exercício: basta determinar o *tamanho* de uma coleção disjunta máxima.) Qual o consumo de tempo do seu algoritmo?

**Exercício 19.B**
Prove que o algoritmo guloso para o problema dos intervalos disjuntos está correto. (Ou seja, prove a propriedade da subestrutura ótima e a propriedade da escolha gulosa.)

**Exercício 19.C** [CLRS 16.1-2]
Mostre que a seguinte idéia também produz um algoritmo guloso correto para o problema da coleção disjunta máxima de intervalos: dentre os intervalos disjuntos dos já escolhidos, escolha um que tenha instante de início máximo. (Em outras palavras, suponha que os intervalos estão em ordem decrescente de início.)

**Exercício 19.D** [CLRS 16.1-4]
Nem todo algoritmo guloso resolve o problema da coleção disjunta máxima de intervalos. Mostre que nenhuma das três idéias a seguir resolve o problema. Idéia 1: Escolha o intervalo de menor duração dentre os que são disjuntos dos intervalos já escolhidos. Idéia 2: Escolha um intervalo seja disjunto dos já escolhidos e intercepte o menor número possível de intervalos ainda não escolhidos. Idéia 3: Escolha o intervalo disjunto dos já selecionados que tenha o menor instante de início.

**Exercício 19.E** [Coloração de intervalos. CLRS 16.1-3]
Queremos distribuir um conjunto de atividades no menor número possivel de salas. Cada atividade $a_i$ ocupa um certo intervalo de tempo $[s_i, f_i)$; duas atividades podem ser programadas para a mesma sala somente se os correspondentes intervalos são disjuntos. Descreva um algoritmo guloso que resolve o problema. (Represente cada sala por uma cor; use o menor número possível de cores para pintar todos os intervalos.) Prove que o número de salas dado pelo algoritmo é, de fato, mínimo.

**Exercício 19.F** [pares de livros]
Suponha dado um conjunto de livros numerados de 1 a $n$. Suponha que o livro $i$ tem peso $p_i$ e que $0 < p_i < 1$ para cada $i$. Problema: acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo 2 livros e o peso do conteúdo de cada envelope seja no máximo 1. Escreva um algoritmo guloso que calcule o número mínimo de envelopes. Mostre que seu algoritmo está correto (ou seja, prove a "greedy-choice property" e a "optimal substructure" apropriadas). Estime o consumo de tempo do seu algoritmo.

**Exercício 19.G** [Bin-packing]
São dados objetos $1, \ldots, n$ e um número ilimitado de "latas". Cada objeto $i$ tem "peso" $w_i$ e cada lata tem "capacidade" 1: a soma dos pesos dos objetos colocados em uma lata não pode passar de 1. Problema: Distribuir os objetos pelo menor número possível de latas.

Programe e teste as seguinte heurísticas. Heurística 1: examine os objetos na ordem dada; tente colocar cada objeto em uma lata já parcialmente ocupada que tiver mais "espaço" livre sobrando; se isso for impossível, pegue uma nova lata. Heurística 2: rearranje os objetos em ordem decrescente de peso; em seguida, aplique a heurística 1. Essas herísticas resolvem o problema? Compare com o exercício 19.F.

Para testar seu programa, sugiro escrever uma rotina que receba $n \leq 100000$ e $u \leq 1$ e gere $w_1, \ldots, w_n$ aleatoriamente, todos no intervalo $(0, u)$.

**Exercício 19.H** [parte de CLRS 16-4, modificado]
Seja $1, \ldots, n$ um conjunto de *tarefas*. Cada tarefa consome um dia de trabalho; durante um dia de trabalho somente uma das tarefas pode ser executada. Os dias de trabalho são numerados de 1 a $n$. A cada tarefa $t$ está associado um *prazo* $p_t$: a tarefa deveria ser executada em algum dia do intervalo $1..p_t$. A cada tarefa $t$ está associada uma *multa* não-negativa $m_t$. Se uma dada tarefa $t$ é executada depois do prazo $p_t$, sou obrigado a pagar a multa $m_t$ (mas a multa não depende do número de dias de atraso). Problema: Programar as tarefas (ou seja, estabelecer uma bijeção entre as tarefas e os dias de trabalho) de modo a minimizar a multa total. Escreva um algoritmo guloso para resolver o problema. Prove que seu algoritmo está correto (ou seja, prove a "greedy-choice property" e a "optimal substructure" apropriadas). Analise o consumo de tempo.