* UM PROBLEMA DE SELEÇÃO DE ATIVIDADES * (Cormen, §16.1)

1. DEFINIÇÃO DO PROBLEMA: a entrada do problema são n >= 1 atividades, a_1 ... a_n, cada a_i começando no tempo c_i e terminando no tempo f_i, os quais são números reais t.q. 0 <= c_i < f_i. Uma solução é um conjunto S ⊆ {1 ... n} tal que, para quaisquer i,j ∈ S, a_i e a_j são compatíveis, isto é, f_i <= c_j ou f_j <= c_i. Uma solução é ótima sse tem tamanho máximo dentre todas as soluções.

2. LEMA: o problema possui subestrutura ótima.

==========

PROVA:

Seja S uma solução ótima para uma entrada a_1 ... a_n. Observe que S não é vazia: como {1} é uma solução, então |S| >= 1. Assim sendo, sejam k ∈ S, A = {i : f_i <= c_k}, D = {i : c_i >= f_k}, AS = S ∩ A e DS = D ∩ S.

Nós mostraremos primeiramente que S = {k} ∪ AS ∪ DS. De fato, se existisse l ∈ S \ ({k} ∪ AS ∪ DS), então, como S é solução, então a_l e a_k seriam compatíveis; logo, ou teríamos f_l <= c_k, e portanto l ∈ AS, um absurdo, ou teríamos c_l >= f_k, e portanto l ∈ DS, um absurdo.

Nós mostraremos agora que AS é solução ótima para a entrada A do problema. De fato, suponha, por RAA, que existe solução B para a entrada A t. q. |B| > |AS|. Como B ⊆ A ⊆ {1 ... n} \ ({k} ∪ D) e DS ⊆ D, então S' = B ∪ {k} ∪ DS é também solução para a entrada a_1 ... a_n e |S'| > |S|, contradizendo a suposição inicial de que S é solução ótima. Logo, AS é ótima para a entrada A, CQD.

De forma análoga, é possível demonstrar que DS é solução ótima para a entrada D, e isso completa a demonstração.

==========

3. UM ALGORITMO DE PROGRAMAÇÃO DINÂMICA P/ O PROBLEMA -- executa em tempo $\theta(n^3)$:

==========

```
Algoritmo: sel_ativ_pd
Entrada: vetores c[1..n] e f[1..n] com os tempos de começo e fim das atividades.
Saída: o tamanho de uma solução ótima.
-------------------------------------------------------------------------------
01. Ordene "c" e "f" com relação aos valores em "f", em ordem crescente.
02. PARA k DE 1 A n
03. | i := k-1
04. | ENQUANTO i >= 1 E f[i] > c[k]
05. | | --i
06. | E[k] := i     // 1a atividade compatível à esquerda
07. | i = k+1
08. | ENQUANTO i <= n E c[i] < f[k]
09. | | ++i
10. | D[k] := i     // 1a atividade compatível à direita
11. PARA i DE 1 A n
12. | T[i,i] := 1
13. PARA t DE 2 A n
14. | PARA i DE 1 a n -t +1
15. | | j := i +t -1 , to := 0
16. | | PARA k DE i A j
17. | | | tk := 1       // Unidade correspondente a a_k
18. | | | SE E[k] >= i ENTÃO tk := tk + T[i,E[k]] // atividades à esquerda
19. | | | SE D[k] <= j ENTÃO tk := tk + T[D[k],j] // atividades à direita
20. | | | SE tk > to   ENTÃO to := tk
21. | | T[i,j] := to   // tamanho ótimo
22. RETORNE T[1,n].
```

==========

4. EXERCÍCIO (INTERESSANTE): o algoritmo acima é baseado no fato de que o problema possui subestrutura ótima, e é basicamente aquele que foi discutido em sala, cuja ideia é: "a melhor solução que existe que possui a atividade a_k é a união de {k} com a melhor solução que existe para o conjunto das atividades que terminam até c_k, e com a melhor solução que existe para o conjunto das atividades que começam a partir de f_k". Em particular, as 3 partes dessa ideia são claramente identificáveis nas linhas 17 a 19.

   Há, entretanto, uma sutileza envolvida. Como discutido em sala, o fato de que as atividades estão dispostas em ordem crescente de finalização garante que o subconjunto das atividades a_i ... a_j que termina até o tempo c_k é exatamente o conjunto {a_i ... a_{E[k]}}: de fato, qualquer atividade que de a_k em diante obviamente não termina até o tempo c_k; além disso, qualquer atividade de a_{E[k] + 1} a a_{k-1} termina depois de c_k, por definição do vetor "E"; por fim, a_{E[k]} termina até c_k (novamente por definição de "E"), e toda atividade anterior a a_{E[k]} não termina depois de a_{E[k]}, dada a ordenação das atividades.

   Observe, entretanto, o que ocorre com as atividades de a_{D[k]} a a_j. Claramente, toda atividade anterior a a_{D[k]} ou não é compatível com a_k ou é compatível mas termina antes de a_k; além disso, a_{D[k]} é compatível com e posterior a a_k. Entretanto, é possível que, dentre as atividades a_{D[k] + 1} a a_j, haja algumas que terminem depois de a_k mas que não sejam compatíveis com a_k. Apesar disso, o algoritmo acima constrói a melhor solução que envolve a atividade a_k com base na melhor solução para a entrada a_{D[k]} ... a_j; por quê?

   (A resposta se encontra mais abaixo nesta nota de aula, mas você deve tentar fazer o exercício por conta própria antes de conferi-la.)

5. LEMA: se a_1 ... a_n é uma entrada cujas atividades estão dispostas em ordem crescente de tempo de finalização, então existe uma solução ótima S tal que 1 ∈ S.

==========
PROVA:

Seja S uma solução ótima para a entrada em questão. Se 1 ∈ S, então o resultado vale. Se 1 ∉ S, seja k = mín(S), e seja T = (S \ {k}) ∪ {1}. Claramente, |T| = |S|; logo, para provarmos o resultado, é suficiente mostrar que T é uma solução para o problema. De fato, como S é uma solução, então todas as atividades em S \ {k} são compatíveis entre si, e portanto resta apenas mostrar que a_1 é compatível com as atividades em S \ {k}. De fato, como S é solução, então cada atividade a_i em S \ {k} é compatível com a_k; além disso, como as atividades estão ordenadas pelo tempo de finalização, e pela definição de k, temos que f_1 <= f_k <= c_i para todo i ∈ S \ {k}. Logo, a_1 é compatível com todas as atividades em S \ {k}, e portanto T é solução, CQD.
==========

6. LEMA: se a_1 ... a_n é uma entrada cujas atividades estão ordenadas em ordem crescente de tempo de finalização, D = {i : c_i >= f_1} e SD é uma solução ótima p/ a entrada ED que é composta pelas atividades de índice em D, então S = SD ∪ {1} é uma solução ótima p/ a entrada E = {a_1 ... a_n}.

==========
PROVA:

Observe primeiramente que S é uma solução para E, pois, como SD é uma solução ótima para a entrada ED, então todas as atividades em ED são compatíveis entre si, e, além disso, pela definição de D, {1} é compatível com todas as atividades em SD.

Resta, então, mostrar não existe solução de E maior que S. De fato, pelo lema anterior, existe uma solução ótima R de E tal que {1} ∈ R. Para toda tal R, como R é solução, então, sendo Q = R \ {1}, temos Q ⊆ D. Finalmente, como SD é solução ótima para ED, então |Q| <= |SD|, e portanto |R| <= |S|, ou seja, não existe solução maior que S, CQD.
==========

7. Resposta para o exercício 4 acima:

==========
PROVA:

Nós mostraremos que, se D[k] <= j, então existe uma solução ótima para a entrada a_{D[k]} ... a_j composta apenas por atividades compatíveis com a_k. De fato, pelo lema 5 acima, existe uma solução ótima S para a entrada em questão tal que D[k] ∈ S. Observe então que toda atividade de uma tal solução S é compatível com a_k, pois esse é o caso para a_{D[k]} (por definição de "D") e, além disso, como toda atividade a_x != a_{D[k]} de S é compatível com a_{D[k]}, então c[x] >= f[D[k]] > c[D[k]] >= f[k] (ou seja, a_x é compatível com a_k). Logo, S tem a propriedade desejada, o que conclui o argumento.
==========

8. UM ALGORITMO GULOSO, baseado no lema 6:

==========
Algoritmo: sel_ativ_gls
Entrada: vetores c[1..n] e f[1..n] com os tempos de começo e fim das atividades.
Saída: o tamanho de uma solução ótima.

```
1. Ordene "c" e "f" com relação aos valores em "f", em ordem crescente.
2. t := 0 , i := 1
3. ENQUANTO i <= n
4. | ++t , l := f[i] , ++i
5. | ENQUANTO i <= n e c[i] < l
6. | | ++i
7. RETORNE t.
```
==========

9. EXERCÍCIOS:

   a) Escreva uma variação do algoritmo acima que utilize apenas um laço.

   b) Estenda o algoritmo acima, de forma que ele retorne uma solução completa para o problema, ao invés de apenas o valor da solução.