

Aula 36: Algoritmo de Huffman

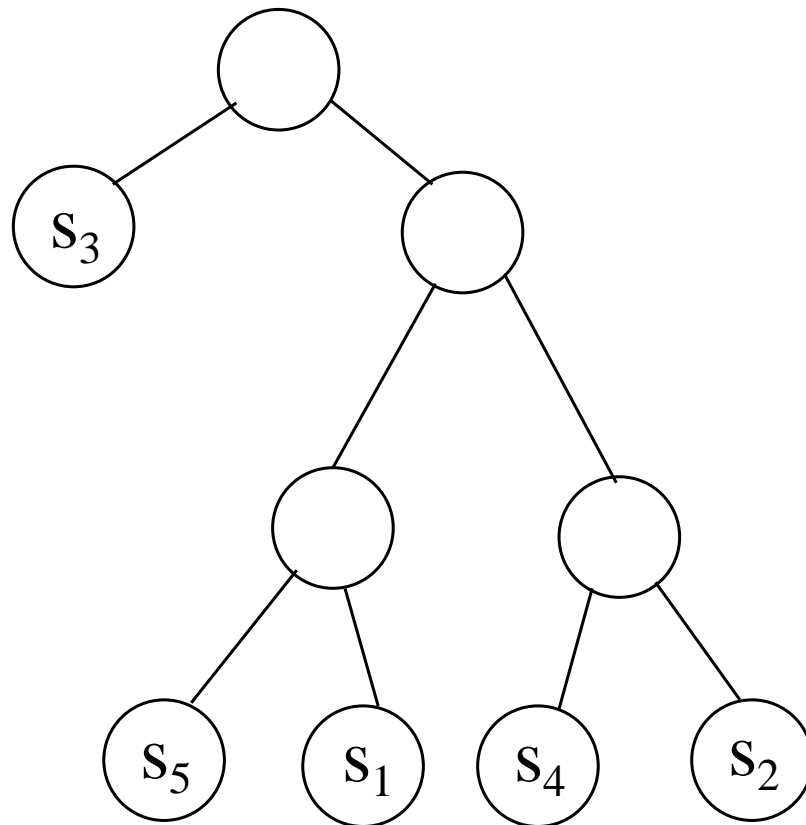
⇒ Descrição do algoritmo

⇒ Correção do algoritmo


Problema

- ➡ Seja $S = \{ s_1, \dots, s_n \}$ um conjunto de elementos, denominados símbolos, cada s_i com uma frequência f_i associada. Construir uma árvore binária de prefixo T para S , de modo a minimizar $\sum f_i l_i$ onde l_i é o comprimento do símbolo s_i em T .
- ➡ T é árvore de Huffman para S

Exemplo



| símbolo | frequência | código |
|---------|------------|--------|
| s_1 | 3 | 101 |
| s_2 | 4 | 111 |
| s_3 | 9 | 0 |
| s_4 | 3 | 110 |
| s_5 | 2 | 100 |



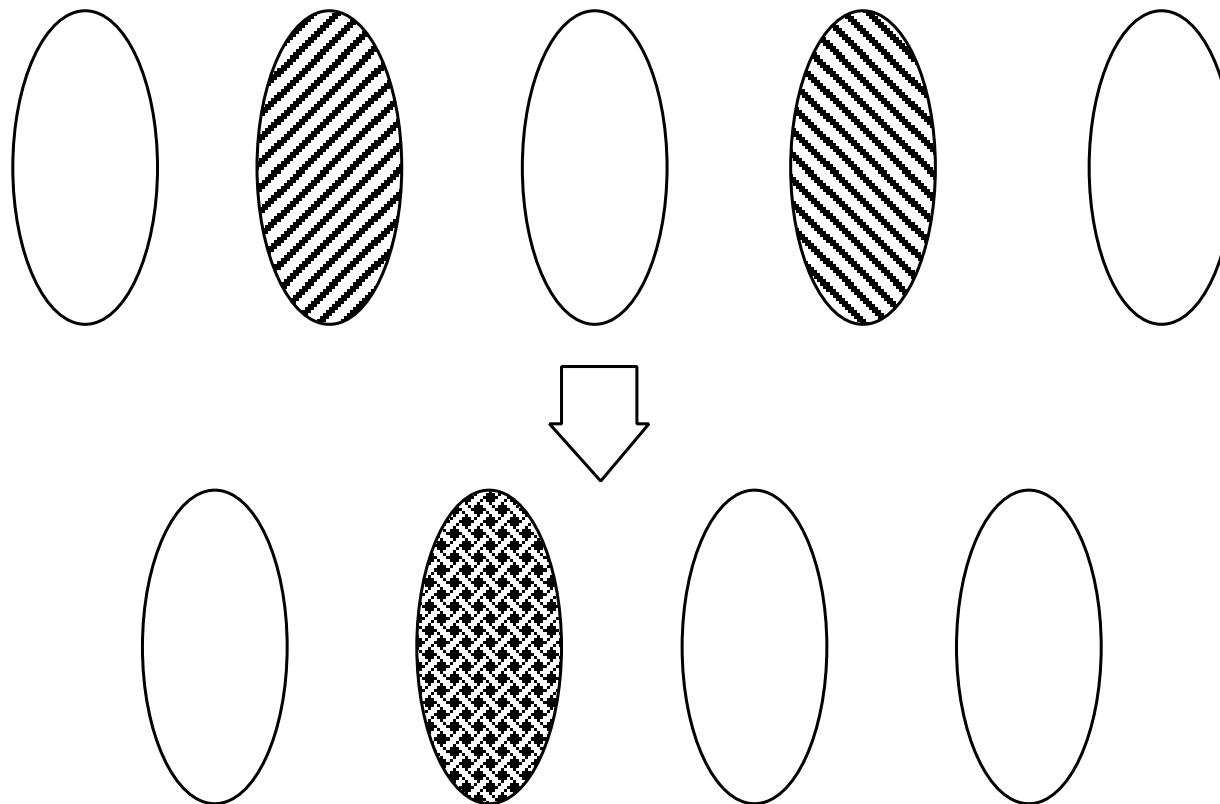
$$c(T) = \sum_{1 \leq i \leq n} f_i l_i = \text{custo de } T$$

Problema (continuação)

- ➡ Para resolver o problema, utiliza-se o algoritmo guloso.
- ➡ O algoritmo constrói a árvore de forma iterativa.
- ➡ A árvore é construída das folhas para a raiz. Isto é, os códigos são construídos de trás para frente.
- ➡ De um modo geral, o processo corresponde a obter subcódigos para subconjuntos de símbolos.
- ➡ Cada um dos subcódigos acima corresponde a uma subárvore.
- ➡ O passo geral iterativo produz a fusão de duas dessas subárvores em uma única.
- ➡ O processo se encerra quando o número de árvores se reduz a um.

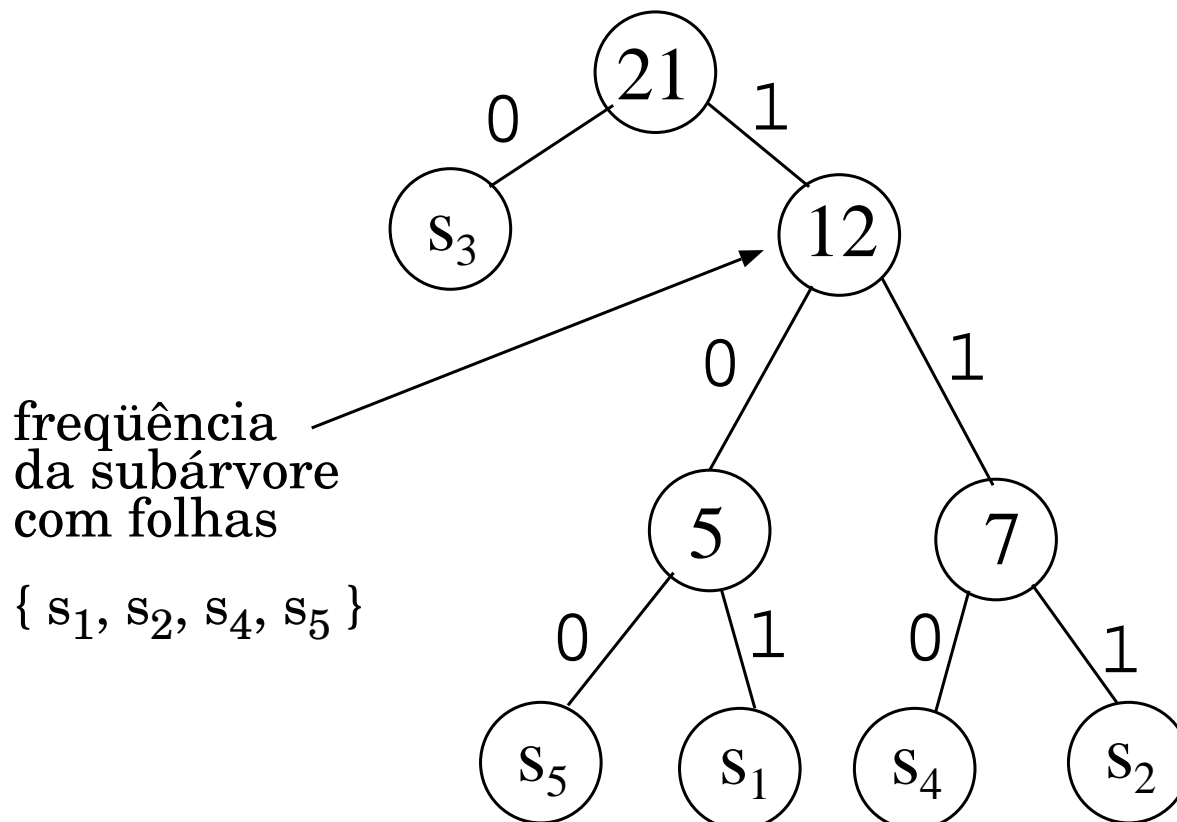
Problema (continuação)

- ➡ O passo geral iterativo produz a fusão de duas dessas subárvores em uma única.
- ➡ O processo se encerra quando o número de árvores se reduz a um.



Frequência de Árvores

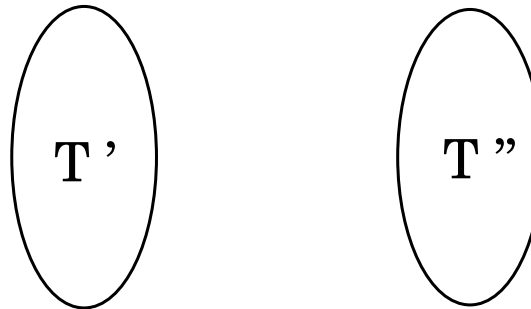
- ➡ T' = subárvore binária de prefixo
- ➡ Cada folha de T' corresponde a um símbolo s_i , com uma frequência f_i .
- ➡ $f(T') = \text{frequência de } T' = \text{soma das frequências dos símbolos nas folhas de } T'$.



| símbolo | frequência |
|---------|------------|
| s_1 | 3 |
| s_2 | 4 |
| s_3 | 9 |
| s_4 | 3 |
| s_5 | 2 |

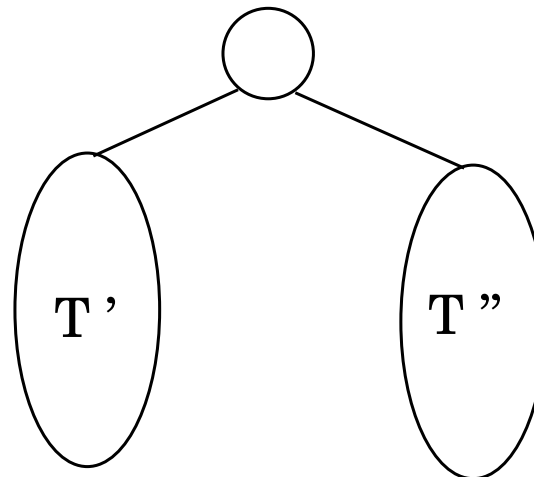
Operação \oplus

⇒ T', T'' = subárvores disjuntas



⇒ $T' \oplus T''$ = subárvore com raiz em um novo nó, e cujas subárvores esquerda e direita são T' e T'' , respectivamente.

⇒ $T' \oplus T''$



Algoritmo de Huffman

➡ Passo inicial:

Definem-se n subárvores, cada qual consistindo de um único nó contendo o símbolo s_i , $1 \leq i \leq n$.

➡ Passo geral:

Repetir $n - 1$ vezes:

Escolher as duas subárvores T' e T'' de menor frequência e substituí-las por $T' \oplus T''$.

Observações sobre o algoritmo

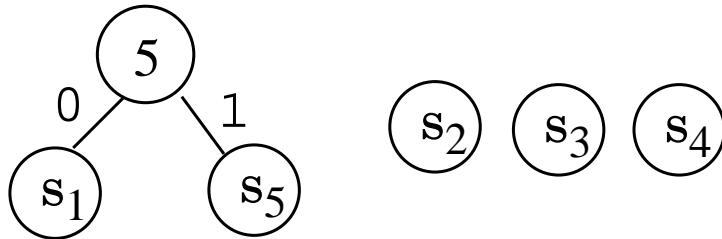
- ⇒ $f(T' \oplus T'') = f(T') + f(T'')$
- ⇒ Em cada iteração do passo geral, o número de subárvores diminui de uma unidade.
- ⇒ Após a última iteração do passo geral, há apenas uma subárvore.
- ⇒ A árvore resultante da última iteração do passo geral é a árvore de Huffman procurada.

Exemplo

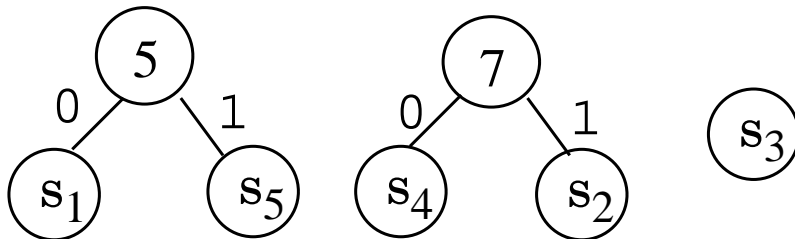
Passo inicial:



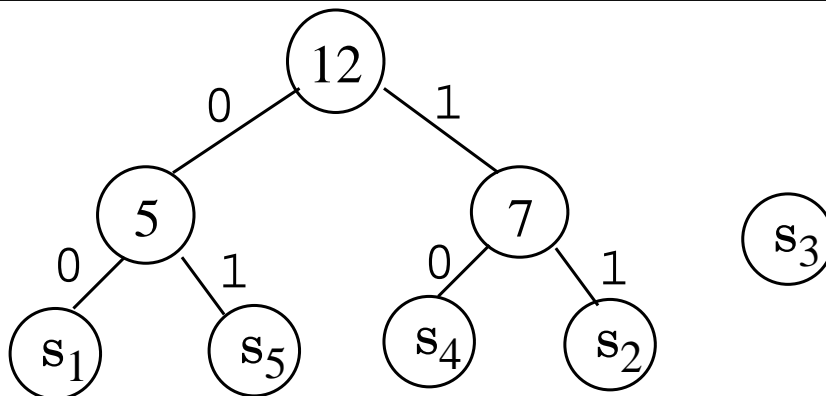
1



2



3



símbolo frequência

s_1 3

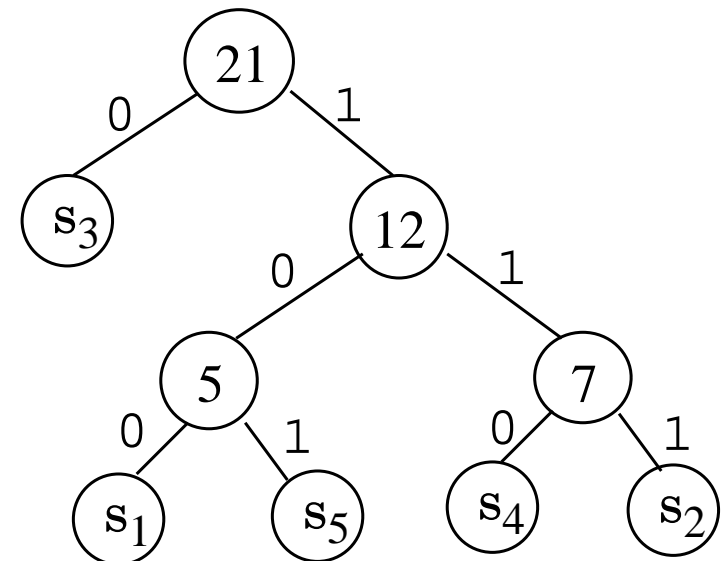
s_2 4

s_3 9

s_4 3

s_5 2

4



Exercício

➡ Desenhar a árvore de Huffman para o seguinte conjunto de símbolos e frequências:

| s_1 | s_2 | s_3 | s_4 | s_5 | s_6 | s_7 | s_8 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| 1 | 6 | 2 | 1 | 1 | 9 | 2 | 3 |

Tempo: 5 minutos

Implementação do algoritmo

➡ Em cada iteração é necessário determinar as duas subárvores T' e T'' de menor frequência. T' e T'' devem ser substituídas por $T' \oplus T''$.

➡ Operações realizadas:

- ▬ minimização
- ▬ inclusão
- ▬ remoção

➡ Estrutura de dados apropriada:

Uma lista de prioridades, por exemplo heap. A ordem das prioridades está invertida, isto é, o elemento a ser buscado e removido da estrutura é aquele de menor prioridade. No caso, as prioridades estão associadas às frequências.

Formulação

➡ Algoritmo: construção da árvore de Huffman

```

para i = 1, ..., n - 1 faça
    mínimo( T ' , F ); mínimo( T ' ' , F );
    T := T ' + T ' '
    f := f ( T ' ) + f ( T ' ' )
    inserir ( T, f, F )
  
```

➡ F = lista de prioridades.

No início, cada nó de F é uma árvore T_i composta de um único nó, com frequência f_i , $i \leq i \leq n$

➡ Mínimo(T ' , F) é um procedimento que determina a árvore T ' de menor prioridade (frequência) de F, e a remove de F.

➡ Inserir(T, f, F) é um procedimento que insere a subárvore T, de prioridade f, em F.

Complexidade

- ➡ Cada operação de minimização ou inclusão na lista de prioridades pode ser efetuada em $O(\log n)$ passos.
- ➡ A operação \oplus requer um número constante de passos.
- ➡ Logo, cada iteração possui complexidade $O(\log n)$.
- ➡ Há um total de $n - 1$ iterações.
- ➡ Na inicialização, são necessários $O(n)$ passos.
- ➡ Complexidade: $O(n \log n)$

Correção do Algoritmo

➡ Lema: sejam símbolos s_i com frequências f_i , $1 \leq i \leq n$, $n > 1$, tais que f_1 e f_2 são as duas menores frequências. Então existe uma árvore de Huffman para esses símbolos, em que os nós s_1 e s_2 , correspondentes a f_1 e f_2 , são irmãos localizados no último nível.

Prova do Lema

⇒ Prova: seja T uma árvore de Huffman para as frequências dadas.

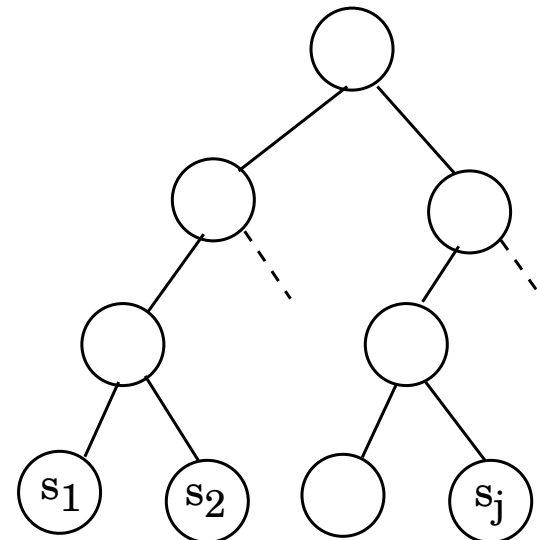
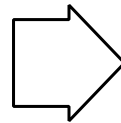
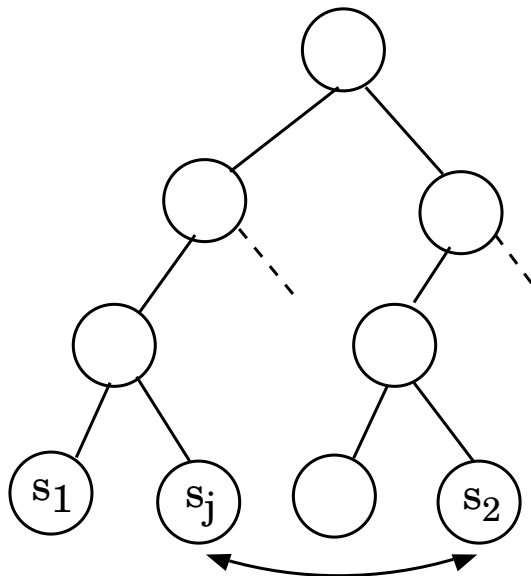
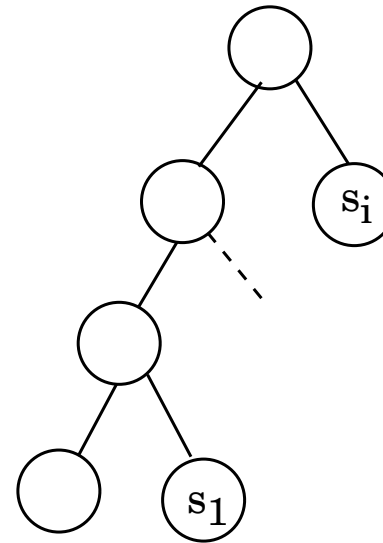
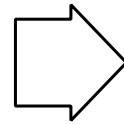
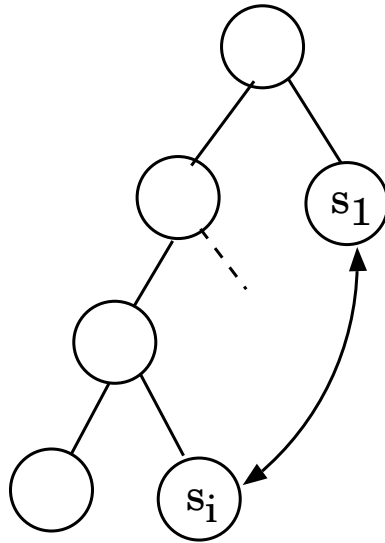
T estritamente binária ⇒ Há pelo menos 2 folhas no último nível.

Se s_1 não está no último nível ⇒ Existe s_i no último nível, onde $f_i = f_1$. Trocar de posição s_1 com s_i .

⇒ Repetir esta operação com s_2 , em lugar de s_1 .

Se s_1 e s_2 não forem irmãos, trocar de posição s_2 com o irmão de s_1 . Ao final, s_1 e s_2 são irmãos localizados no último nível.

Prova do Lema



Correção do Algoritmo

➡ Teorema: Seja T a árvore construída pelo algoritmo de Huffman para as frequências f_1, \dots, f_n , $n > 1$. Então T é mínima.

➡ Prova: Seja $f_1 \leq \dots \leq f_n$

T_{\min} = árvore ótima para f_1, \dots, f_n .

Indução em n . Se $n = 2$, trivial.

Suponha $n > 2$. Pela hipótese de indução, o algoritmo de Huffman sempre obtém uma árvore de custo mínimo quando o número de símbolos é menor que n .

Examine o algoritmo para f_1, \dots, f_n . No primeiro passo, são eliminadas as subárvores com as frequências f_1 e f_2 e substituídas por uma subárvore com frequência $f_1 + f_2$.

Correção do Algoritmo (cont.)

➡ Seja T' a árvore construída pelo algoritmo para $f_1 + f_2, f_3, \dots, f_n$.
Pela hipótese de indução, T' é mínima. Logo,

$$c(T) = c(T') + f_1 + f_2 \quad (i)$$

Por outro lado, seja T'' a árvore obtida de T_{\min} eliminando-se as folhas (nós irmãos) correspondentes a f_1 e f_2 , e associando ao pai delas um novo símbolo, com frequência $f_1 + f_2$. T'' é uma árvore binária de prefixo correspondente às frequências $f_1 + f_2, f_3, \dots, f_n$. Logo,

$$c(T_{\min}) = c(T'') + f_1 + f_2 \quad (ii)$$

Comparando (i) e (ii) e observando que $c(T') \leq c(T'')$, conclui-se que $c(T) = c(T_{\min})$ e T é uma árvore mínima.

Exercícios Finais

➡ Descrever um algoritmo para determinar a árvore de Huffman relativa a um conjunto de símbolos e frequências dadas, que possua altura mínima.

➡ Dado um conjunto de n arquivos A_1, \dots, A_n , ordenados, o problema da intercalação de arquivos consiste em reuni-los em um único arquivo ordenado. Para tal, um programa padrão intercala os arquivos dois a dois. É necessário, portanto, executar o programa de intercalação um total de $n-1$ vezes.

Sabe-se que cada arquivo A_i possui $|A_i|$ chaves e que para intercalar os arquivos A_i e A_j o programa utiliza $|A_i| + |A_j|$ comparações entre chaves. Determinar a ordem em que as intercalações devem ser realizadas de modo a minimizar o número total de comparações efetuadas.