

Projeto de algoritmos: Algoritmos Gulosos

ACH2002 - Introdução à Ciência da Computação II

Delano M. Beder

Escola de Artes, Ciências e Humanidades (EACH)
Universidade de São Paulo
dbeder@usp.br

09/2008

- Tipicamente algoritmos gulosos são utilizados para resolver problemas de **otimização**.
- Um algoritmo guloso sempre faz a **escolha** que parece ser a **melhor** a cada iteração.
- Ou seja, a **escolha** é feita de acordo com um **critério guloso** !
 - É uma decisão localmente ótima !
- Propriedade da **escolha gulosa**: garante que a cada iteração é tomada uma decisão que levará a um ótimo global.
- Em um algoritmo guloso uma escolha que foi feita **nunca é revista**, ou seja, não há qualquer tipo de *backtracking*.

Algoritmos Gulosos: Idéias Básicas

- A idéia básica da **estratégia gulosa** é construir por etapas uma solução **ótima**.
- Em cada passo, após selecionar um elemento da entrada (o melhor), decide-se se ele é viável - vindo a fazer parte da solução - ou não.
- Após uma seqüência de decisões, uma solução para o problema é alcançada.
- Nessa seqüência de decisões, nenhum elemento é examinado mais de uma vez: ou ele fará parte da solução, ou será descartado.
- Frequentemente, a entrada do problema já vem ordenada.

Intercalação (Fusão) de Vetores

No livro [2], foi apresentado um método para intercalação(fusão) de vetores:

Intercalação de Vetores

```
int [] intercalacao(int [] a, int [] b) {
    int posa = 0,
        posb = 0,
        posc = 0;
    int [] c = new int [a.length + b.length];

    // Enquanto nenhuma das seqüências está vazia...
    while (posa < a.length && posb < b.length) {
        // Pega o menor elemento das duas seqüências

        if(b[posb] <= a[posa]) {
            c[posc] = b[posb];
            posb++;
        } else {
            c[posc] = a[posa];
            posa++;
        }
        posc++;
    }
}
```

Intercalação de Vetores (continuação)

```
// Completa com a sequência que ainda não acabou

while (posa < a.length) {
    c[posc] = a[posa];
    posc++;
    posa++;
}

while (posb < b.length) {
    c[posc] = b[posb];
    posc++;
    posb++;
}

return c; // retorna o valor resultado da intercalação
}
```

- Um exemplo simples que ilustra bem a estratégia gulosa é o problema da intercalação sucessiva de vetores.
 - Trata-se de intercalar n vetores
- Há várias maneiras possíveis de se realizar uma sequência de intercalações
 - Ocorre que isso pode afetar o número de comparações necessárias
- Lembremos o caso mais simples de intercalação de dois vetores, com tamanhos m_1 e m_2
 - O tamanho do vetor resultante é $m_1 + m_2$
 - O número de comparações necessárias (no pior caso) será $m_1 + m_2 - 1$

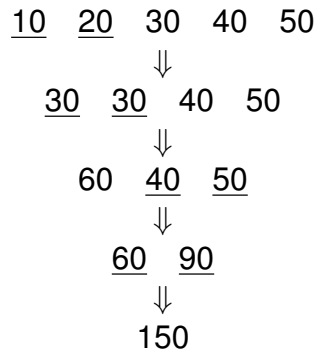
- Três vetores V_1 , V_2 e V_3 , com respectivos tamanhos 15, 10 e 5.
- Uma maneira de realizar a intercalação seria: primeiro intercalar V_1 e V_2 e depois o vetor resultante V_{12} com V_3 .
 - O número de comparações para produzir V_{12} é $15 + 10 - 1 = 24$.
 - Para intercalar V_{12} com V_3 é $(15 + 10) + 5 - 1 = 29$.
 - No total, teremos $24 + 29$, ou seja, 53 comparações.
- Primeiro intercalar V_2 e V_3 e depois o vetor resultante V_{23} com V_1 .
 - O número de comparações para produzir V_{23} é $10 + 5 - 1 = 14$.
 - Para intercalar V_{23} com V_1 é $(10 + 5) + 15 - 1 = 29$.
 - No total, teremos $14 + 29$, ou seja, 43 comparações.
- Primeiro intercalar V_1 e V_3 e depois o vetor resultante V_{13} com V_2 . Quantas comparações são feitas ?
- A ordem das intercalações influencia substancialmente o número de comparações requeridas

Quantidade de comparações

- $V_1 + V_2 \Rightarrow V_{12}$ e $V_{12} + V_3 \Rightarrow V_{123}$
 - $m_1 + m_2 - 1$ para produzir V_{12}
 - $m_{12} + m_3 - 1$ para produzir V_{123}
 - Ou seja, $m_1 + m_2 - 1 + m_1 + m_2 + m_3 - 1 \Rightarrow 2(m_1 + m_2) + m_3 - 2$
- $V_2 + V_3 \Rightarrow V_{23}$ e $V_{23} + V_1 \Rightarrow V_{123}$
 - $m_2 + m_3 - 1$ para produzir V_{23}
 - $m_{23} + m_1 - 1$ para produzir V_{123}
 - Ou seja, $m_2 + m_3 - 1 + m_2 + m_3 + m_1 - 1 \Rightarrow 2(m_2 + m_3) + m_1 - 2$
- $V_1 + V_3 \Rightarrow V_{13}$ e $V_{13} + V_2 \Rightarrow V_{123}$
 - $m_1 + m_3 - 1$ para produzir V_{13}
 - $m_{13} + m_2 - 1$ para produzir V_{123}
 - Ou seja, $m_1 + m_3 - 1 + m_1 + m_3 + m_2 - 1 \Rightarrow 2(m_1 + m_3) + m_2 - 2$

O primeiro par de vetores a ser intercalado dá contribuição com maior peso para o total.

- um algoritmo guloso para esse problema
 - Seleciona, a cada passo um par com soma mínima dentre os presentes para fazer parte da solução.
 - Ao final, teremos uma sequência ótima de intercalações.



```

procedimento intercala_ótima_vetores
Entrada: Conjunto V de n vetores (com seus tamanhos)
Saída: par (intercalação dos n vetores, número de comparações)
BEGIN
  numCmp = 0;
  REPITA
    | escolhe os dois menores vetores A e B (seleção gulosa) |
    V = V - { A , B };
    C = Intercala ( A , B );
    V = V + { C };
    numCmp = numCmp + tam(A) + tam(B) - 1
  ATÉ QUE |V| = 1;
  retorne_saída(V, numCmp);
END
    
```

```

int[] merge(int[][] conjunto) {

  int tam = conjunto.length;
  int numCmp = 0;

  do {
    /* escolhe os dois menores vetores A e B (seleção gulosa) */
    Menores menores = menoresVetores(conjunto);
    int prim = menores.getPrimeiro();
    int seg = menores.getSegundo();
    int[] A = conjunto[prim];
    int[] B = conjunto[seg];
    /* V = V - A, B ; */
    conjunto = removeVetores(conjunto, menores);
    /* C = Intercala(A, B); */
    int[] C = merge(A, B);
    /* V = V + C */
    conjunto[tam - 2] = C;
    numCmp = numCmp + A.length + B.length - 1;
    tam = tam - 1;
  } while (tam > 1);
  System.out.println("Foram feitas " + numCmp + " Comparações");
  return conjunto[0];
}
    
```

Resumo

Foi apresentada a técnica de projeto de algoritmos: *algoritmos gulosos*

- A idéia básica da **estratégia gulosa** é construir por etapas uma solução **ótima**.
 - Em cada passo, após selecionar um elemento da entrada (o melhor), decide-se se ele é viável - vindo a fazer parte da solução - ou não.

Referências utilizadas

- [1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein. *Algoritmos - Tradução da 2a. Edição Americana*. Editora Campus, 2002.
- [2] Laira V. Toscani & Paulo A.S. Veloso. *Complexidade de Algoritmos*. Série Livros Didáticos, Instituto de Informática da UFRGS, 2a. Edição, 2005.