Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Algoritmos e Estruturas de Dados
6 de Novembro de 2007
Segunda série de exercícios, Semestre de Inverno de 06/07.
Data limite de entrega: 26 de Novembro de 2007

---

1 Tipos Abstractos de Dados

Pretende-se realizar uma implementação eficiente do tipo de dados abstracto *map*, que representa um contentor associativo de pares *chave-valor*, em que os elementos estão ordenados de forma crescente pelo valor da chave. Não são admitidos elementos com chaves repetidas ou nulas. A interface deste contentor é representada pela seguinte `interface` em Java:

```
public interface Map<Key extends Comparable<Key>, Value> {
   public boolean isEmpty(); // Testa se está vazio.
   public Value search(Key key); // Devolve o valor associado à chave ou null caso este não exista.
   public boolean insert(Key key, Value value); // Insere par chave-valor, retornando true.
                                                // Caso a chave já exista ou seja null, retorna false.
   public boolean remove(Key key); // Remove par chave-valor, retornando true.
                                   // Caso a chave não exista, retorna false.
}
```

A estrutura de dados adoptada deve ter as seguintes características:

- Composta por N listas, L_0 a L_{N-1}, simplesmente ligadas e ordenadas, em que N é definido aquando da construção da estrutura de dados. A lista L_i é designada por lista de nível i.
- Todos os elementos contidos na estrutura de dados devem estar presentes na lista L_0. Se um elemento está presente na lista L_i, então também tem de estar presente nas listas $L_0, \ldots, L_{i-1}$. O nível de um elemento é definido como sendo o maior nível das listas a que pertence.
- Qualquer elemento é representado por apenas um nó, mesmo que esteja presente em mais do que uma lista. Para tal, o campo *next* deve ser um *array* de referências para nó.

A figura 1 esquematiza esta estrutura de dados.

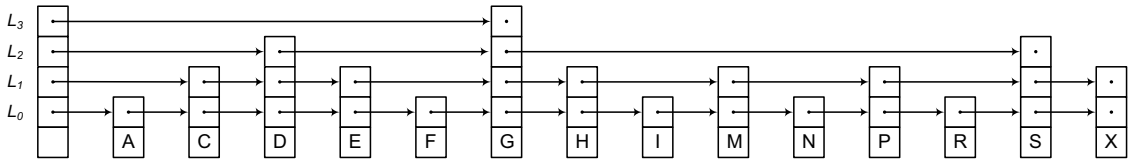

Figura 1: Esquema da estrutura de dados a implementar.

O algoritmo de pesquisa deve ser uma variante do algoritmo binário de pesquisa, baseado na seguinte observação: sejam k a chave do elemento a procurar e n_i e n_j dois nós consecutivos da lista L_m, para qualquer m entre 0 e $N-1$, tais que $n_i.k < k < n_j.k$ ($n.k$ denota a chave do nó n); se o elemento de chave k existir então está localizado entre os nós n_i e n_j.

No algoritmo de inserção, o nível do elemento deve ser gerado de forma pseudo-aleatória, usando o método apresentado em seguida.

```
private static int newLevel(int maxLevel) {
   int level, j;  double t = Math.random();
   for (level = 1, j = 2; level < maxLevel; ++level, j += j)
      if (t*j > 1.0)  break;
   return level;
}
```

O algoritmo de inserção usa a estratégia da procura para a determinação do local de inserção.

A figura 2 ilustra um exemplo de procura e inserção de um novo nó.

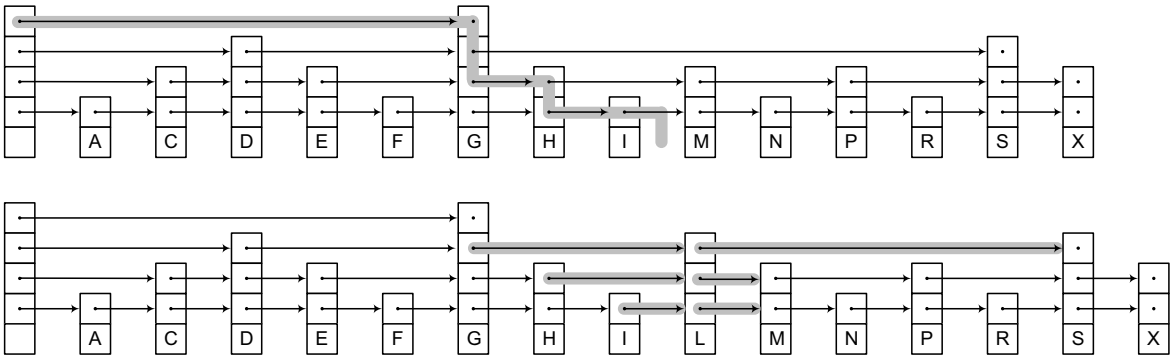

Figura 2: Procura e inserção na lista.

2 Notação O e recorrências

2.1. Sejam $f, g : \mathbf{N}_0 \longrightarrow \mathbf{R}^+$ duas funções.

a) Recorrendo apenas à definição de O, prove que $O(f) \cdot O(g) = O(f \cdot g)$.

b) Sejam $f(n)$ e $g(n)$ definidas por: $f(n) = n$ e $g(n) = 2n + 3$.
Prove que os conjuntos $O(f)$ e $O(g)$ são iguais.

2.2. Resolva as seguintes recorrências, apresentando o resultado sob a forma de uma função explícita de n.

a) $C(n) = C(n/2) + n^2$, para $N \geq 2$ com $C(1) = 0$, quando n é uma potência de 2.

b) $C(n) = C(n/\alpha) + 1$, para $n \geq 2$ com $C(1) = 0$, quando n é uma potência de α.

3 Algoritmos recorrentes

Desenvolva implementações recorrentes dos seguintes métodos:

3.1. `public void reverse()` da classe `SLinkedList<E>`, que inverte a ordem dos elementos da lista.

3.2. `public static int union(int[] a1, int[] a2, int[] res)` da classe `ArrayUtils`, que realiza a *união* de dois conjuntos de números inteiros, representados pelos *arrays ordenados* `a1` e `a2`. O resultado da união é colocado no *array* `res` passado em argumento. Assuma que `res` é construído previamente com dimensão suficiente para albergar o resultado da união. O método retorna o número de elementos efectivos do conjunto união.

3.3. `public static int[][] spiral(int n)` da classe `ArrayUtils`, que devolve uma matriz, de dimensões $n \times n$ ($n \in \mathbf{N}$), preenchida com os valores de 1 a n^2, com a seguinte configuração (exemplo para $n = 4$):

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Analise a complexidade, no pior caso, dos algoritmos implementados resolvendo as equações de recorrência associadas.