Filas de prioridade e heapsort

- Além do Sedgewick (**sempre** leiam o Sedgewick), veja
  - `http://www.ime.usp.br/~pf/algoritmos/aulas/hpsrt.html`

Filas de prioridade e heapsort

- *Fila de prioridade*: fila na qual a operação de remoção atende um critério de *prioridade*
- Filas de prioridade podem ser usadas para ordenar itens: basta que a noção de prioridade seja compatível com a noção de ordem usada para a ordenação desejada [isto é, prioridade máxima = objeto de maior/menor chave]
- Se tivermos uma fila de prioridade na qual inserções e remoções podem ser feitas em tempo $O(\log N)$ (supondo que manipulamos no máximo N objetos), então teremos um algoritmo de ordenação com complexidade de tempo $O(N \log N)$!

Filas de prioridade

```
/* prog9.1.c - PQ.h */
void PQinit(int);
 int PQempty();
void PQinsert(Item);
Item PQdelmax();
```

Implementações elementares

Alternativas:

- Vetor ordenado
- Lista ordenada
- Vetor não-ordenado
- Lista não-ordenada

Implementações elementares

Vetor não-ordenado:

```
/* prog9.2.c */
#include <stdlib.h>
#include "Item.h"

static Item *pq;
static int N;

void PQinit(int maxN)
  { pq = malloc(maxN*sizeof(Item)); N = 0; }
 int PQempty()
  { return N == 0; }
```

Implementações elementares

```
/* prog9.2.c */
[...]
void PQinsert(Item v)
  { pq[N++] = v; }
Item PQdelmax()
  { int j, max = 0;
    for (j = 1; j < N; j++)
      if (less(pq[max], pq[j])) max = j;
    exch(pq[max], pq[N-1]);
    return pq[--N];
  }
```

Implementação com “heaps”

Consideramos coleções de itens com chave, organizados em forma de *árvore binária*, com um item por nó da árvore.

Definição 1. *Uma tal árvore binária está organizada na forma de heap se, para todo nó, a chave do item naquele nó é maior ou igual à chave dos itens nos filhos deste nó.*

Naturalmente, em uma árvore organizada na forma de heap, a chave na raiz tem valor máximo.

Implementação com “heaps”

Podemos representar *árvores binárias completas* com vetores, de forma natural: o pai de um nó na posição $k > 1$ do vetor está na posição $\lfloor k/2 \rfloor$ do vetor. A raiz está na posição 1; a posição 0 é ignorada.

Definição 2. *Um heap é uma árvore como na Definição 1, representada em um vetor como acima.*

Algoritmos sobre heaps

```
/* prog9.3.c */
void fixUp(Item a[], int k)
  {
    while (k > 1 && less(a[k/2], a[k]))
      { exch(a[k], a[k/2]); k = k/2; }
  }
```

Algoritmos sobre heaps

```
/* prog9.4.c */
void fixDown(Item a[], int k, int N)
  { int j;
    while (2*k <= N)
      { j = 2*k;
        if (j < N && less(a[j], a[j+1])) j++;
        if (!less(a[k], a[j])) break;
        exch(a[k], a[j]); k = j;
      }
  }
```

Complexidade dos algoritmos para heaps

Propriedade 3. *Suponha que temos um heap com* N *itens.*

- ▷ *A inserção de um* $(N+1)$*-ésimo elemento pode ser feita executando no máximo* $\log_2(N+1)$ *comparações entre itens.*
- ▷ *A operação de remoção de máximo pode ser feita executando no máximo* $2\lfloor \log_2(N-1) \rfloor$ *comparações entre itens.*