Trabalho T2

Fila de prioridade

(implementada como heap binário)

Izabella Magalhães Paulette

Eduardo Radaelli Coppetti

Fabio Correa Costa Junior

Fila de Prioridade:

Uma fila de prioridade se compara com uma fila comum, porem a sua ordem é dada pela prioridade de cada elemento que a compõe, ou seja, o primeiro elemento a sair da fila de prioridade é o elemento que conter a prioridade mais alta.

Entre as funções usadas em filas de prioridades, duas delas são consideradas funções básicas para a implementação:

1. Inserção; e
2. Remoção.

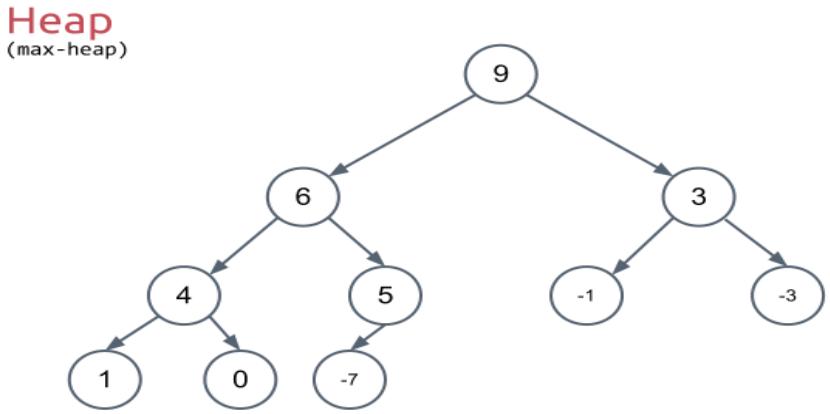
Além de várias funções usadas na implementação também existem várias estruturas capazes de formar/ser usada como fila de prioridade, entre elas está a estrutura de heap binário.

Heap:

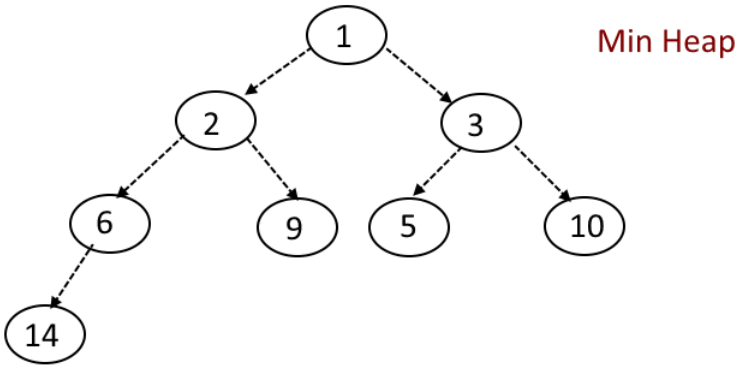
A estrutura de heap é especializada, baseada na de arvores. É uma estrutura considerada bastante eficiente para uso em fila de prioridade.

Essa estrutura pode ser usada de duas formas:

- A máx. heap ou heap máxima, nessa forma os nós pai são maiores ou iguais aos nós filhos; e



- A min heap ou heap mínima, nessa forma os nós pai são menores ou iguais aos nós filhos.



Porem ambos tem os nós pai tendo sua prioridade mais alta que a dos nós filhos, sendo a raiz de maior prioridade.

Geralmente a estrutura de heap é implementada em array e não necessita do uso de ponteiros entre seus elementos.

Heap Binário:

Enquanto o heap é baseado em arvores a heap binária é baseada em árvores binárias. Chamada principalmente de árvore binária completa, pois todos os níveis da árvore, exceto possivelmente o último (mais profundo) são totalmente preenchidos e se o último nível não estar completo, os nós desse nível devem ser preenchidos da esquerda para a direita.

E assim como a estrutura heap, a heap binária mantém a forma de max heap e min heap.

Operações de filas de prioridades como heap binaria:

- *Inserção*: Adiciona um novo nó ao heap.
- *Remoção*: Remove o elemento da raiz do heap; em geral, troca-se o elemento na última posição com a raiz, diminui-se a referência à última posição armazenada e então são feitas alterações de forma a preservar os invariantes do *heap*.
- Heapsort: Ordena os nós em que sempre o menor ou maior elemento deve estar na raiz, para ordenar os dados.

Exemplos de Filas de Prioridades:

- Filas de bancos;
- Filas de emergência em hospitais;
- Filas de espera para transplante de órgãos;
- Fila de processos.

Algoritmos basicos da implementação de fila de prioridade como heap binario:

- Struct:

```c
#define MAX 100

struct dados{
    int info;
    int prioridade;
};
typedef struct dados Dados;

struct fila_prioridade{
    int quantidade;
    Dados dados[MAX];
};
typedef struct fila_prioridade FilaPrio;
```

- Inserção:

A função insere coloca um novo elemento na heap.

```c
int insere_FilaPrio(FilaPrio* fp, int info, int prio){
    if(fp==NULL){ //testa existencia da fila de prioridade
        return 0;
    }
    if(fp->quantidade==MAX){ //verifica se a fila de prioridade esta cheia
        printf("Fila de priorirade lotada\n");
        return 0;
    }
    int aux;
    fp->dados[fp->quantidade].info=info;
    fp->dados[fp->quantidade].prioridade=prio;
    aux=promoverElemento(fp,fp->quantidade);
    fp->quantidade++;
    return 1;
}
```

- Remoção:

A função remove retira o primeiro elemento(raiz) da heap.

```c
int remove_FilaPrio(FilaPrio* fp){
    if(fp==NULL){ //testa existencia da fila de prioridade
        return 0;
    }
    if(fp->quantidade==0){ //verifica se a fila de prioridade esta vazia
        printf("Fila Prio esta vazia\n");
        return 0;
    }
    int aux;
    fp->quantidade--;
    fp->dados[0]=fp->dados[fp->quantidade];
    aux=rebaixaElemento(fp,0);
    return 1;
}
```

- Ordenação:

Para ordenar a heap podem ser usadas duas funções:

1. A função promove elemento é usada para ordenar o elemento recém inserido. Quando inserido o elemento é colocado no fim da heap e precisa ser encaixado nas propriedades de fila de prioridade.

```
int promoverElemento(FilaPrio* fp, int filho){
    int pai;
    Dados temp;
    pai=(filho-1)/2;
    while((filho>0)&&(fp->dados[pai].prioridade<=fp->dados[filho].prioridade)){
        temp= fp->dados[filho];
        fp->dados[filho]=fp->dados[pai];
        fp->dados[pai]=temp;
        filho=pai;
        pai=(pai-1)/2;
    }
    return 0;
}
```

2. A função rebaixa elemento é usada quando o primeiro elemento(raiz) é removido da heap. Quando o elemento(raiz) é removido, é colocado o último elemento(folha) com menor prioridade da estrutura no lugar da primeira, deixando a fila de prioridade desordenada. Então a função ordena a heap novamente.

```
int rebaixaElemento(FilaPrio* fp, int pai){
    Dados temp;
    int filho=2*pai+1;
    while(filho<fp->quantidade){
        if(filho<fp->quantidade-1){
            if(fp->dados[filho].prioridade<fp->dados[filho+1].prioridade)
                filho++;
            }
        }
        if(fp->dados[pai].prioridade>=fp->dados[filho].prioridade){
            break;
        }
        temp=fp->dados[pai];
        fp->dados[pai]=fp->dados[filho];
        fp->dados[filho]=temp;
        pai=filho;
        filho=2*pai+1;
    }
    return 0;
}
```

- Consulta:

Na função consulta é retornado o primeiro elemento da fila de prioridade.

```
int consulta_FilaPrio(FilaPrio* fp, int primeiro){
    if(fp==NULL || fp->quantidade==0){//verifica se a fila de prioridade existe ou esta vazia
        return 0;
    }
    primeiro=fp->dados[fp->quantidade-1].info;
    return primeiro;
}
```

Referências Bibliográficas:

Estruturas de Dados Usando C (PDF)
>https://www.cin.ufpe.br/~garme/public/(ebook)Estruturas%20de%20Dados%20Usando%20C%20(Tenenbaum).pdf

MC-202 Filas de Prioridade e Heap (PDF)
>https://www.ic.unicamp.br/~rafael/cursos/2s2018/mc202/slides/unidade21-fila-de-prioridade.pdf

Fila de Prioridade (PDF)

>http://www.facom.ufu.br/~abdala/DAS5102/TEO_HeapFilaDePrioridade.pdf

[ED] Aula 85 - Fila de Prioridades: Definição
>https://www.youtube.com/playlist?list=PL8iN9FQ7_jt5ld45Ed-Nm8-eMme7Y5rEo

Heap
>https://pt.wikipedia.org/wiki/Heap

A estrutura heap
>https://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/heap.html

Aula por Lucca Siaudzionis
>https://noic.com.br/materiais-informatica/curso/aula-16-estruturas-de-dados-iii-heap-fila-de-prioridade/

Heap.c

>http://www.each.usp.br/digiampietri/ACH2023/javaec/Heap/heap.c

Fila de Prioridade / Siang Wun Song - Universidade de São Paulo - IME/USP

>https://www.ime.usp.br/~song/mac5710/slides/03prior.pdf