priority_queue.a

優先權佇列使用說明書

611121212 李奕承

摘要

本library提供的結構如下

```
typedef struct HeapType {
    void * elements;
    int numElementds;
}Heap_t;

typedef enum {
    MINHEAP = 0,
    MAXHEAP
}H_class;

typedef struct PQ {
    H_class pqClass;
    Heap_t heap;
    int maxSize;
    int elementSize;
    int (*compare)(void * elementA, void * elementB);
}PQ_t;
```

本 library 提供以下幾種函數 - 產生優先權佇列函數 - 判斷優先權佇列是否為空函數 - 以完全二元樹印出優先權佇列函數 - 插入優先權佇列函數 - 計算優先權佇列Level函數 - 判斷優先權佇列是否已滿函數 - 取出優先權佇列函數

使用步驟

1.帶入標頭檔

```
#include"pqSpec.h"
```

2.創造priority queue

宣告一個 PQ_t ，使用 createPQ() 填入PQ_t的各項設定

```
PQ_t first_pq;
createPQ(&first_pq, MINHEAP, 10, compareMath);
```

3.編譯

- pqSpec.h 要記得放在引用標頭檔的路徑下
- 編譯時記得加入 priority_queue.a

```
gcc -I include -Wall -o test test.c priority_queue.a
```

函數介紹與範例

範例使用的結構

下面是範例用到的結構

```
typedef struct test_element {
    char ID[10];
    int math;
    int eng;
}student_t;
```

下面是範例中使用者提供給 libary 用來比較 priority 的函數

```
int compareMath(void * elementA, void * elementB)
{
    if(((student_t *)elementA)->math > ((student_t *)elementB)->math)
    {
        return 1;
    }
    else if(((student_t *)elementA)->math < ((student_t *)elementB)->math)
    {
        return -1;
    }
    return 0;
}
```

下面範例中使用者用來取出元素中數值的函數

```
char * get_math(void * element)
{
    static char buffer[20];
    sprintf(buffer, "%d", ((student_t *)element)->math);
    return buffer;
}
```

產生優先權佇列函數

```
void createPQ(PQ_t * pq, H_class pqClass, int maxSize,
              int (*compare)(void * elementA, void * elementB));
```

當你宣告一個 PQ_t 之後，要使用這個函數來設定 PQ_t 中需要用到的值

其中包括 PQ_t 的地址、Heap的類型(最大或最小)、priority queue 容納的上限、用來比較 priority 的函數(由使用者提供)，以下是範例。

```
PQ_t first_pq;
createPQ(&first_pq, MINHEAP, 10, compareMath);
```

判斷優先權佇列是否為空函數

```
int IsEmpty(PQ_t * pq);
```

使用這個函數可以判斷 priority queue 是否為空，以下是範例。

```
printf("is queue empty? 0->true 1->false : %d\n", IsEmpty(&first_pq));
```

終端機的輸出

```
is queue empty? 0->true 1->false : 0
```

以完全二元樹印出優先權佇列函數

```
void pq_printf_tree(PQ_t * pq,
                    char * (*get_data)(void * queue_member));
```

使用這個函數可以將 priority queue 以完全二元樹的形式印出來，便於除錯，以下是範例

```
pq_printf_tree(&first_pq, get_math);
```

終端機的輸出可以在待會看到

插入優先權佇列函數

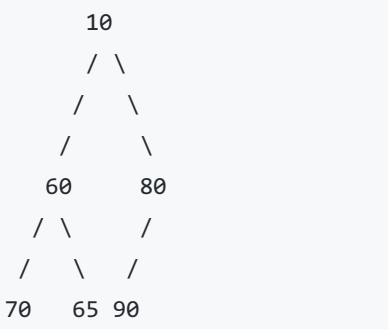
```
int Enqueue(PQ_t * pq, void * elementA);
```

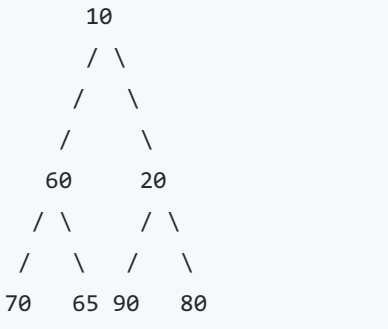
使用這個函數可以在 priority queue 中加入新的元素，以下是範例。

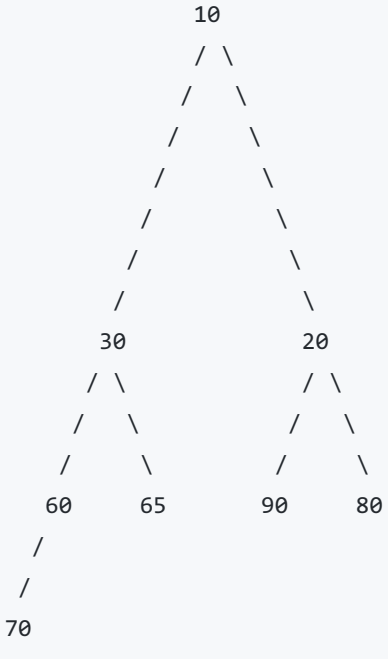
```
 student_t node[10] = {
    {"A", 70, 100},
    {"B", 60, 90},
    {"C", 80, 95},
    {"D", 65, 90},
    {"E", 10, 70},
    {"F", 90, 90},
    {"G", 20, 60},
    {"H", 30, 50},
    {"I", 40, 40},
    {"J", 50, 30}
};

for(int i = 0 ; i < 10 ; i++)
{
    Enqueue(&first_pq, &node[i]);
    pq_printf_tree(&first_pq, get_math);
}
```

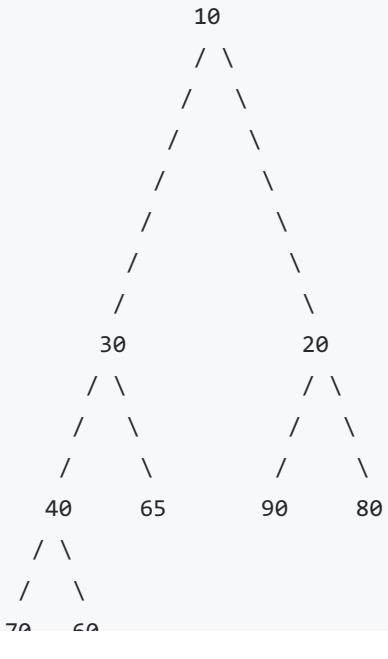
終端機的輸出

```
  70

   60
   /
  /
 /
70


   60
   / \
  /   \
 /     \
70     80


       60
       / \
      /   \
     /     \
    /       \
   65       80
  /
 /
70


       10
       / \
      /   \
     /     \
    /       \
   60       80
  / \
 /   \
70   65
```


10
60 80
70 65 90


10
60 20
70 65 90 80


10
30 20
60 65 90 80
70


10
30 20
40 65 90 80
70

```
70   60

              10
              / \
             /   \
            /     \
           /       \
          /         \
         /           \
        /             \
       30             20
      / \             / \
     /   \           /   \
    /     \         /     \
   40     50       90     80
  / \     /
 /   \   /
70   60 65
```

計算優先權佇列Level函數

```
int count_pq_level(PQ_t * pq);
```

使用這個函數可以算出 priority queue 的Level數，以下是範例。

```
pq_printf_tree(&first_pq, get_math);
printf("the tree level is : %d\n", count_pq_level(&first_pq));
```

```
              10
              / \
             /   \
            /     \
           /       \
          /         \
         /           \
        /             \
       30             20
      / \             / \
     /   \           /   \
    /     \         /     \
   40     50       90     80
  / \     /
 /   \   /
70   60 65

the tree level is : 3
```

判斷優先權佇列是否已滿函數

```
int IsFull(PQ_t * pq);
```

使用這個函數可以判斷 priority queue 是否已滿，以下是範例。

```
printf("is queue full? 0->true 1->false : %d\n", IsFull(&first_pq));
```

終端機的輸出

```
is queue full? 0->true 1->false : 0
```

取出優先權佇列函數

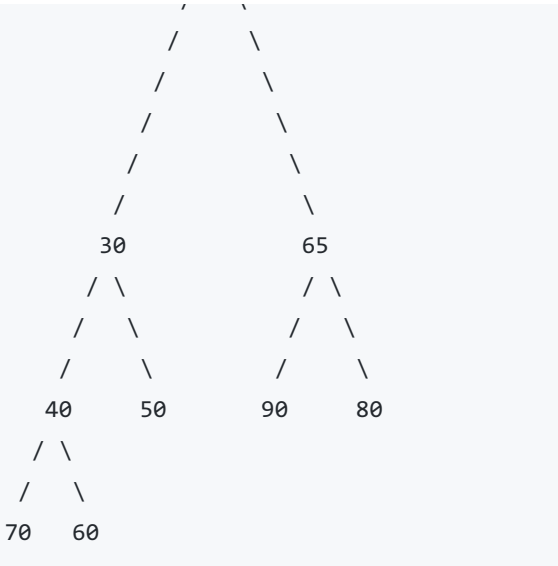
```
void * Dequeue(PQ_t * pq);
```

使用這個函數可以在 priority queue 中取出最上層的元素，以下是範例。

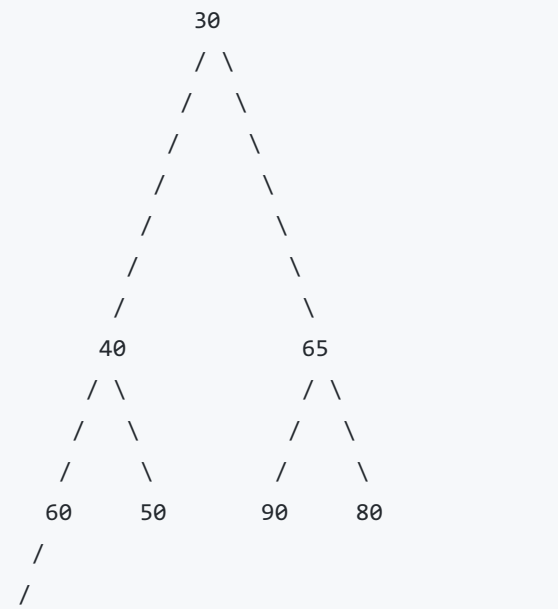
```
pq_printf_tree(&first_pq, get_math);

for(int i = 0 ; i<10 ; i++)
{
    printf("----------------------------------------\n");
    printf("get : %s\n", get_math(Dequeue(&second_pq)));
    pq_printf_tree(&second_pq, get_math);
}
```

終端機的輸出

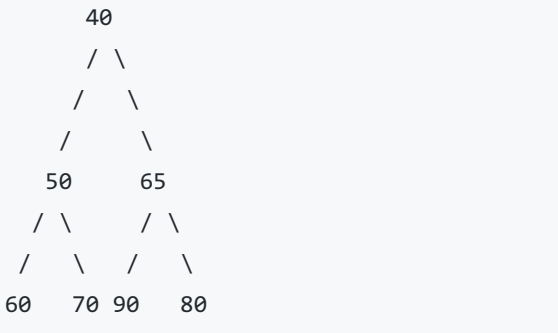
```
              10
             / \
            /   \
           /     \
          /       \
         /         \
        /           \
       /             \
      30             20
     / \             / \
    /   \           /   \
   /     \         /     \
  40     50       90     80
 / \     /
/   \   /
70   60 65
----------------------------------------
get : 10

              20
             / \
            /   \
```



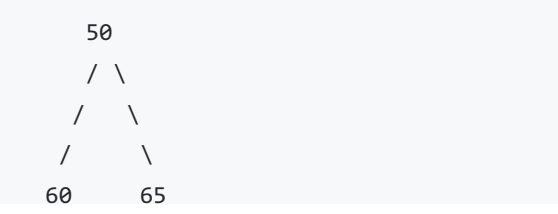
```
----------------------------------------
get : 20
```



```
----------------------------------------
get : 30
```



```
----------------------------------------
get : 40
```



```
  / \     /
 /   \   /
80   70 90
----------------------------------------
get : 50

      60
      / \
     /   \
    /     \
   70     65
  / \
 /   \
80   90
----------------------------------------
get : 60

      65
      / \
     /   \
    /     \
   70     90
  /
 /
80
----------------------------------------
get : 65

  70
  / \
 /   \
80   90
----------------------------------------
get : 70

  80
  /
 /
90
----------------------------------------
get : 80

 90
----------------------------------------
get : 90
```