acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Алгоритмы» . BinaryHeapCPP :

Реализация бинарной кучи на C++ и C

Здесь приведена реализация структуры данных "Бинарная куча" на языках программирования C и C++.

Один из алгоритмов, в котором используется бинарная куча – это HeapSort – сортировка методом бинарной кучи, которая в худшем и среднем случае сортирует за время O(N log N) (напомним, что QuickSort в худдем случае работает N^2).

HeapSort реализован в функции main.

Код на C++

```
#include<stdio.h>

typedef struct {
 // int value;
    int  key;
} ITEM;

class HEAP {
public:
    ITEM *h;
    int  size;

    HEAP(unsigned int n) {
       size = 0;
       h = (ITEM*) malloc( sizeof(ITEM) * n);
    }

    ~HEAP() {
        if(h) free(h);
    }

    int add(ITEM x) {
       h[++size]=x;
       checkup(size);
       return 1;
    }

    int extract_min(ITEM *x) {
      if(size ==0) return 0;
      *x = h[1];
      h[1] = h[size--];
      checkdown(1);
      return 1;
    }

private:
    void checkup(int c) {
        int p;
        p = c / 2;
        if( p == 0 )return;
        if(h[p].key > h[c].key) {
           ITEM tmp;
           tmp = h[p]; h[p] = h[c]; h[c] = tmp;
           checkup(p);
        }
    }
```

Поиск
Поиск
Раздел «Алгоритмы»
Главная
Форум
Ссылки
El Judge
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
    void checkdown(int p) {
        int c;
        c = 2*p;
        if( c > size ) return;
        if( c+1 <= size && h[c + 1].key < h[c].key ) c++;

        if( h[c].key < h[p].key ) {
          ITEM tmp;
          tmp =  h[c]; h[c] = h[p]; h[p] = tmp;
          checkdown(c);
       }
    }
};

int main() {
    HEAP heap(1000);
    int n, i;
    ITEM x;

    scanf("%d", &n);

    for(i = 0; i < n; i++){
       scanf("%d", &x.key);
       heap.add(x);
    }

    while( heap.extract_min(&x) ) {
       printf("%d ", x.key);
    }

    return 0;
}
```

Код на C

Ниже приведен более "индустриальный код", соответствующий стандартам. В нём реализовано динамическое выделением памяти.

Заложена возможность динамического увеличения размера кучи.

```
#include<stdio.h>
#include<malloc.h>
typedef int key_t;
typedef unsigned int value_t;

typedef struct {
   key_t  key;
   value_t value;
} pair_t ;

typedef struct {
    pair_t *data;
    unsigned int size;
    unsigned int data_size;
} bheap_t;

bheap_t*
bheap_new( unsigned int initial_data_size ) {
    bheap_t *h = (bheap_t*) malloc( sizeof(bheap_t) );
    h->data = (pair_t*) malloc( sizeof(pair_t) * (1 + initial_data_size) );
    h->data_size = 1 + initial_data_size;
    h->size = 0;
    return h;
}

void
bheap_delete(bheap_t *h) {
    if( h ) {
        if( h->data ) {
```

```
            free( h->data );
        }
        free(h);
    }
}

void
bheap_checkup( bheap_t *h, unsigned int c ) {
    unsigned int p;
    for( p = c / 2;  p > 0 ; c = p , p = c / 2 ) {
        if( h->data[p].key > h->data[c].key ) {
            pair_t tmp = h->data[p]; h->data[p] = h->data[c]; h->data[c] = tmp;
        } else {
            break;
        }
    }
}

void
bheap_checkdown( bheap_t *h, unsigned int p ) {
    unsigned int c;
    for( c = 2 * p ;  c <= h->size  ; p = c, c = 2 *p ) {
        if( c + 1 <= h->size && h->data[c + 1].key < h->data[c].key ) c++;
        if( h->data[c].key < h->data[p].key ) {
            pair_t  tmp;
            tmp =  h->data[c]; h->data[c] = h->data[p]; h->data[p] = tmp;
        } else {
            break;
        }
    }
}


void
bheap_add( bheap_t *h, pair_t v ) {
    if( h->size + 1 >= h->data_size ) {
        h->data_size *= 2;
        h->data = (pair_t*) realloc( h->data, h->data_size * sizeof(pair_t) );
    }
    h->data[ ++(h->size) ] = v ;
    bheap_checkup( h, h->size );
}

int
bheap_extract_min(bheap_t *h, pair_t *v) {
    if( h->size == 0 ) return 0;
    *v = h->data[1];
    h->data[1] = h->data[ (h->size)--];
    bheap_checkdown( h, 1 );
    return 1;
}

int main() {
    bheap_t *h = bheap_new(1000);
    unsigned int n, i;
    pair_t v = {0, 0};

    scanf("%u", &n);
    for( i = 0; i < n; i++ ){
        scanf("%d", &v.key);
        v.value = i;
        bheap_add(h, v);
    }

    while( bheap_extract_min(h, &v) ) {
        printf("%d ", v.key);
    }
    printf("\n");
    bheap_delete( h );
```

```
    return 0;
}
```