Zadania domowe. Blok 3. Zestaw 3

Maciej Poleski

11 maja 2012

1 Scalanie kilku tablic

Tablice zostaną umieszczone na kopcu typu minimum. Dane wejściowe są w `input[m]`. Zakładam że dla każdej tablicy wejściowej mam swobodny dostęp do jej początku i końca oraz że możemy porównać dowolne dwa elementy z dowolnych dwóch tablic wejściowych.

```
template<typename T>
struct HeapEntry
{
    T *iterator;
    T *end;

    bool operator<(const HeapEntry &o) const
    {
        return *iterator<*(o.iterator);
    }
};

template<typename T>
T* solution()
{
    Heap<HeapEntry<T>> heap;
    T* result=new T[n];
    T* resultIterator=result;
    for(auto array : input)
    {
        heap.push(array.begin, array.end);
    }
    while(!heap.isEmpty())
    {
        auto entry=heap.pop();
        *resultIterator++=*(entry.iterator)++;
        if(entry.iterator!=entry.end)
        {
            heap.push(entry);
        }
    }
    return result;
}
```

`T` jest typem elementów w tablicach wejściowych. Algorytm można zoptymalizować modyfikując kopiec - pozwalając na modyfikacje elementów na kopcu (ponieważ elementy w tablicach są posortowane jest to kwestia wykonania `downheap` po usunięciu pierwszego elementu z tablicy).

2 Równanie rekurencyjne

Po podstawieniu do wzoru otrzymujemy:

$$d = \log_{\sqrt{n}} \sqrt{n} = 1$$

$$T(n) = \Theta(n \log n)$$