MC3305: Partial Quicksort

Due on Tuesday, June 9, 2015

Jesus P. Mena-Chalco

Rodrigo Martins de Oliveira, Victor Campagner de Barros

Problem 1

In this report, we're going to talk about partial quicksort.

In the last lectures, we were presented to partial sorting algorithms, which are based on traditional sorting algorithms where only the first k elements of an entry vector are sorted.

Quicksort is the most used sorting algorithm even though it is O(n^2) in its worst case scenario. It is considered the best one by many because its average case is the fastest O(n.$logn$) between sorting by comparison.

Quicksort not only is efficient but also very simple. Its first step is choosing a pivot, which can be chosen however. Next, the rest of the vector is split into higher than pivot and lower or equal than pivot, each of these are called recursively and all the steps are repeated until a call has a part of the vector with size 1.

Partial quicksort uses the recursive calls in its advantage, running quicksort until the pivot is higher than k, and when it is, the right recursive call is ignored and only the left part is called.

Partial Quicksort

```
    void partial_quicksort(int *vet, int esquerda, int direita, int k) {
    int i, j;
    int x, y;
    i = esquerda;
    j = direita;
    if(k > direita) x = vet[(esquerda + direita) / 2];
    else x = vet[(esquerda + k) / 2];

    while(i <= j) {
        while(vet[i] < x && i < direita) i++;
        while(vet[j] > x && j > esquerda) j--;
        if(i <= j)  {
            y = vet[i];
            vet[i] = vet[j];
            vet[j] = y;
            i++;
            j--;
        }
    }
    if(j > esquerda) partial_quicksort(vet, esquerda, j, k);
    if(i < direita && j-esquerda < k-1) partial_quicksort(vet,  i, direita, k);
}
```