LINIER-TIME SORTING AND ORDER STATISTICS

Bucket Sort
Radix Sort
Randomized-Select
Selection in linier time

Bucket Sort

- Uniformly allocate inputs to “buckets” – input must be uniform
- Apply insertion sort to each bucket – O(N^2), but ideal for small inputs because small constant factor
- Combine buckets
- Code (for 0<doubles <1):
- Create n buckets (e.g. vectors)
- --for i to n:
  - Insert element i into bucket (int)(element*n)
- Insertion sort each bucket (or bucket sort)
- Concatenate and print
- This is a stable sort if the insertion sort implementation is stable
- Runs in O(n) average case

Radix Sort

- Useful when sorting an element by a set of properties where there is a strict order of importance for each property and the number of possible states of each property is small: e.g. the digits of an integer, sorting integers.
- Radix sort:
- -for each property, in order of least importance:
  - Use a linier stable sort (e.g. bucket sort) on the property for all elements

  Takes O(pk) time, where p is the number of properties and k is the time taken by the inner-loop sort: p is generally a constant, and k=O(n).

Randomized-Select

- Find's the kth largest number in A[1…n]
- Randomized partition function: Chooses a random pivot as in quicksort, and reorders the elements around the pivot such that those smaller than it are on the left, and larger on the right: O(N)

```
int random_pivot(int a, int b, vector<int> &A){
    int pivot=a+rand()%(b-a+1);
    int i=a;
    s(pivot,b,A);//switch pivot with end element

    for(int j=i;j<b;j++){
        if(A[j]<=A[b]){
            s(i,j,A);
            i++;
        }
    }
    s(i,b,A);
    return i;
}
```

Randomized Select Code:

```
int random_select(int a, int b,int k, vector<int> &A){
    int pivot=random_pivot(a,b,A);
    if(pivot==k){
        return A[k];
    }else{
        if(pivot>k){
            return random_select(a,pivot-1,k,A);
        }else{
            return random_select(pivot+1,b,k,A);
        }
    }
}
```

O(N) average case

Performance (CLOCKS/CLOCKS PER SEC)

- N=10^4
- 0.003 0.008
- N=10^5
- 0.032 0.117
- N=10^6
- 0.077 1.285
- 0.134 1.369
- 0.206 1.265
- N=10^7
- 2.117 16.879
- 1.827 13.924
- 1.584 14.072

Guaranteed Linier Time Selection

- Guarantee a “good” split for partition:
- Group all elements of A into groups of 5
- Use insertion sort to determine median of each group
- Use select function to find the median of medians i.e element (n/5)/2 of the median set
- Partition with this median as the pivot, and call select to the left or right accordingly