Important Algorithms Notebook for IOI

Nebhrajani A.V.

December 3, 2020

Contents

1 Introduction **2**

2 Searchers **2**

2.1 Binary search 2

1 Introduction

Most problems can be reduced to either one of or a combination of the algorithms presented in this notebook. While not meant to be as comprehensive as a set of notes on CLRS, it includes the more challenging and important programs. Sorters and searchers (with the exception of binary search, which I struggled with) are omitted since they're basic and C++'s STL implements them well enough. This notebook focuses majorly on graph algorithms, dynamic programming, greedy programs, and trees. It assumes familiarity with C/C++ and basic programming techniques such as arrays/vectors, loops, lists, iteration, and recursion.
Note that it is bad practice to use the header `bits/stdc++.h`: it is **not** a standard header, and increases compile time by including *every* C++ library, not only the required ones. However, it's okay in competitive programming since the programs are small and runtime matters more than compile time. Obviously, if compilation takes too long, the bottleneck is probably this header.

2 Searchers

2.1 Binary search

```cpp
// To search a sorted array for a value.
#include <bits/stdc++.h>

int binary_search(std::vector<int>& a, int k, int l, int r)
{
  if (r >= l) {
    int mid = (l+r)/2;
    if (k == a[mid]) {
      return mid;
    }
    if (a[mid] < k) {
      return binary_search(a, k, mid+1, r);
    }
    if (a[mid] > k) {
      return binary_search(a, k, l, mid-1);
    }
  }
  return -1;
}

int main()
{
  std::vector<int> arr = {34,54,23,35,13,35,46,678,34,576,46,34,646};
  std::sort(std::begin(arr), std::end(arr));
  int find = binary_search(arr, 46, 0, arr.size());
  std::cout << arr[find] << "\n";
  return 0;
}
```

A basic algorithm, with a lot of similarity to QuickSort. r and l are the indexes of the subarray to search recursively. If $r < l$ directly return -1, the element wasn't found. If not, see if the current *mid* is the element we're looking for. If so, return *mid*. Otherwise, recurse on either the LHS or RHS subarray.