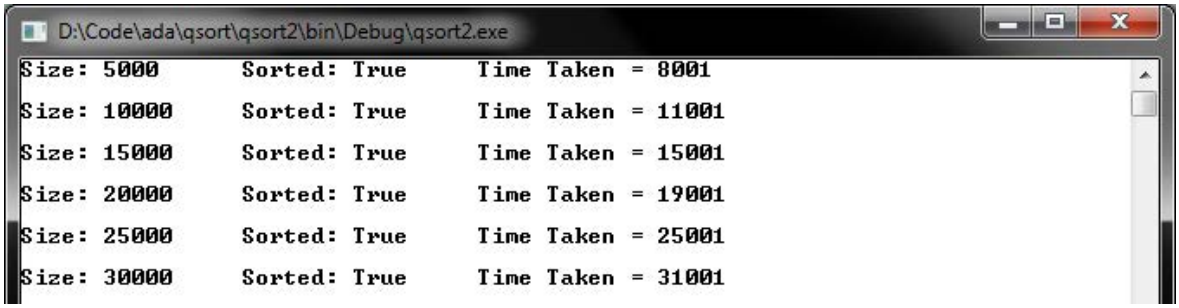
Quick Sort

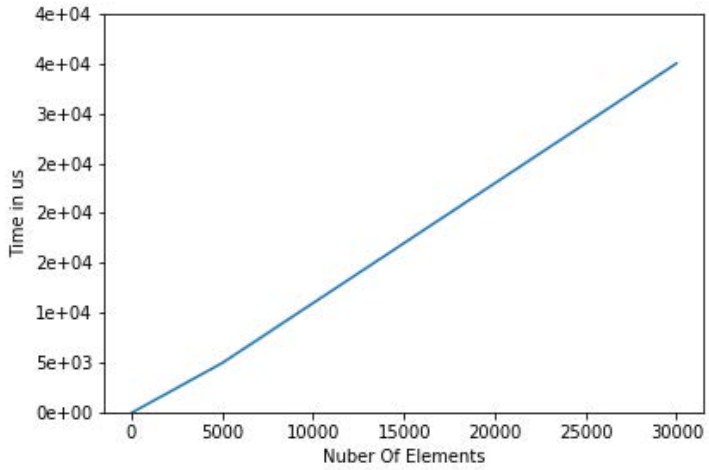
```
#include <algorithm>
#include <iostream>
#include <vector>
#include <random>
#include <time.h>
#define LL long long
using namespace std;

mt19937 rnd;
uniform_int_distribution<uint64_t> dist(0,1e4);
LL gen() {
  return dist(rnd);
}

void qsort(vector<LL>& V, LL start, LL end) { //Both inclusive
  if(start >= end) {
    return;
  }
  LL pindex = gen()%(end-start) + start;

  //Swap pindex and end
  iter_swap(V.begin()+pindex, V.begin()+end);

  //Partition using pindex as pivot
  LL idx=start-1;
  for(LL i=start; i<end; i++) {
    if(V[i] < V[end]) {
      idx++;
      iter_swap(V.begin()+i, V.begin()+idx);
    }
  }

  //put pivot back
  idx++;
  iter_swap(V.begin()+idx, V.begin()+end);

  //subcalls
  qsort(V, start, idx-1);
  qsort(V, idx+1, end);
}
```

```
int main()
{
   vector<LL> time;

   for(LL n=1; n<=30001; n+=5000)
   {
      vector<LL> V(n,0);
      //Average Case: generate(V.begin(), V.end(), gen);
      //Best Case:
      for(LL i=0; i<n; i++) {
         V[i] = n-i;
      }
      //Worst Case: use n-i instrad of i
      starttime = getTime();
      qsort(V, 0, V.size()-1);
      endtime = getTime();
      time.push_back(endtime-starttime);
      cout<<"Size: "<<n-1<<"\tSorted: "<<(is_sorted(V.begin(), V.end()) == 1? "True\t":"False\t"); printTime();
cout<<endl<<endl;
   }
   cout<<"\n[";
   LL n=0;
   for(LL i=0; i<time.size(); i++) {
      cout<<"["<<n<<","<<time[i]<<"],";
      n+=5000;
   }
   cout<<"\b]";

   return 0;
}
```

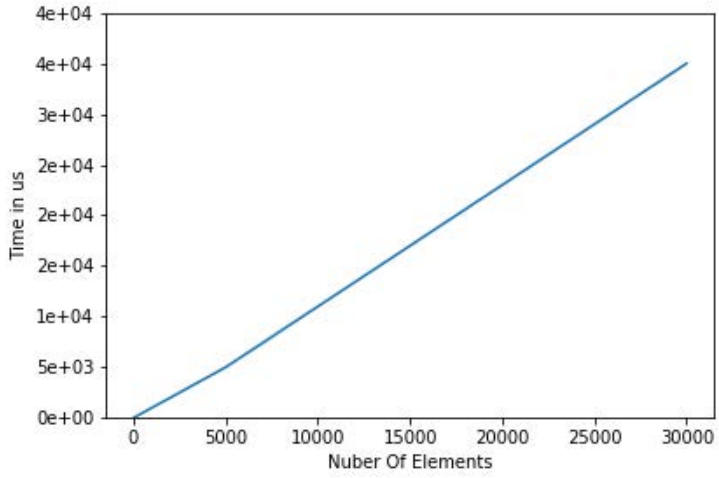
Output:



Average Case:



Best Case:



Worst Case: