Counting Sort

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

template <class T> ostream& operator<<(ostream& out, vector<T> &A) {
   for(LL i=0; i<A.size(); i++) {
      out<<A[i]<<" ";
   } out<<endl;
   return out;
}


vector<LL> countingSort(vector<LL> &A)
{
   vector<LL> B, C;
   LL maxv = -1e17;
   for(LL i=0; i<A.size(); i++) {
      maxv = max(maxv,A[i]);
   }

   B.resize(maxv+1);
   C.resize(A.size());

   //Populate B with A
   for(LL i=0; i<A.size(); i++) {
      B[A[i]]++;
   }
   //Create a cumulative frequency
   for(LL i=1; i<B.size(); i++) {
      B[i] += B[i-1];
   }
   //Fill Values in C
   for(LL i=A.size()-1; i>=0; i--) {
      C[B[A[i]]-1] = A[i];
      B[A[i]]--;
   }
   return C;
}
```

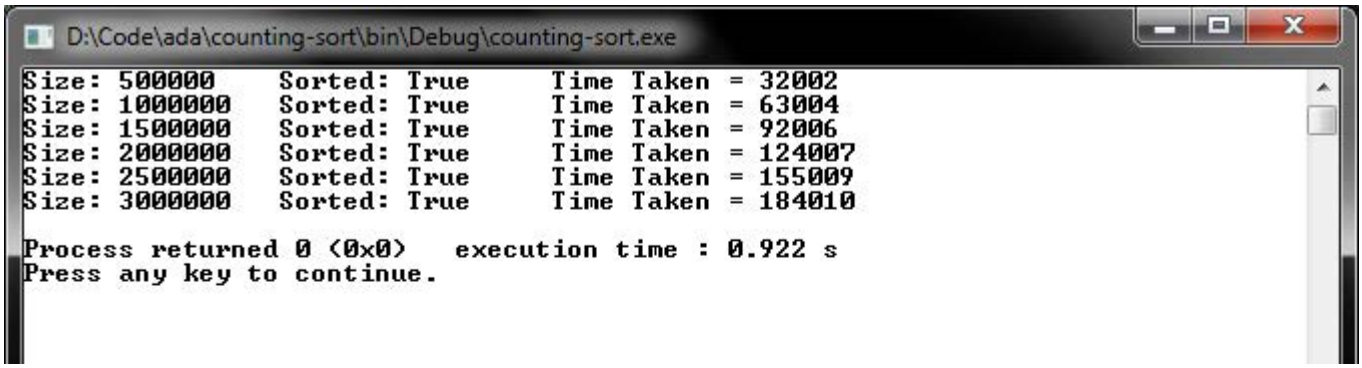
```
mt19937 rnd;
uniform_int_distribution<uint64_t> dist(0,1e4);
LL gen() {
   return dist(rnd);
}

int main()
{
   vector<LL> time;
   LL N = 3000001, sz = 500000;
   for(LL n=1; n<=N; n+=sz)
   {
      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();
      V = countingSort(V);
      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;

   }

   return 0;
}
```

Output:



Time Taken: