# Assignment 1: Introduction to Systems Programming

Francis Vo

October 2, 2012

## 1 Mathematical Analysis

## 2 Theoretical Correctness

## 3 Testing

## 4 Experimental Analysis

## 5 Extrapolation and Interpretation

## 6 Code

### 6.1 Algorthm 1

```
/*
 * Enumeration
 * Loop over each pair of indices i; j and compute the sum from k=i
     to j of a[k].
 * Keep the best sum you have found so far.
 */

#include <iostream>
#include <cstdio>
#include <stdio.h>
#include <stdlib.h>

int MaxSubarray(int a[], int n);

int main(int argc, char **argv){
  int input[] = {31, -41, 59, 26, -53, 58, 97, -93, -23, 84};
  std::cout << MaxSubarray(input,10) << std::endl;
}

int MaxSubarray(int a[], int n){

  int i,j,k;
  int max = a[0];
  int sum;
  for(i = 0; i < n; ++i ){
    for (j = i; j < n; ++j){
```

```
      sum = 0;
      for (k = i; k <=j; ++k){
        sum += a[k];
      }
      if(max < sum){
        max = sum;
      }
    }
  }
  return max;
}
```

alg1.cpp

## 6.2 Algorthm 2

```
/*
 * Better Enumeration
 * Notice that in the previous algorithm, the same sum is computed
     many times.
 * In particular, notice that sum from k=i to j of a[k] can be
     computed from sum from k=i to j - 1 of a[k] in O(1) time,
     rather than starting from scratch.
 * Write a new version of the frst algorithm that takes advantage
     of this observation.
 */

#include <iostream>
#include <cstdio>
#include <stdio.h>
#include <stdlib.h>

int MaxSubarray(int a[], int n);

int main(int argc, char **argv){
  int input[] = {31, -41, 59, 26, -53, 58, 97, -93, -23, 84};
  std::cout << MaxSubarray(input,10) << std::endl;
}

int MaxSubarray(int a[], int n){

  int i,j,k;
  int max = a[0];
  int sum;
  for(i = 0; i < n; ++i ){
    sum = 0;
    for (j = i; j < n; ++j){
      sum += a[j];
      if(max < sum){
        max = sum;
      }
    }
  }
  return max;
}
```

alg2.cpp

## 6.3 Algorthm 3

```
/*
 * Divide and Conquer
 * If we split the array into two halves, we know that the maximum subarray will either be
 *       * contained entirely in the frst half,
 *       * contained entirely in the second half, or
 *       * made of a suffix of the frst half of maximum sum and a prefix of the second half of maximum sum
 * The frst two cases can be found recursively. The last case can be found in linear time.
 */
```

alg3.cpp