A picture containing text

Description automatically generated

*Prof. Emdad Khan*

*September 2019*

*Lab#1*

***Group 1***

*Group members:*

*Asad Ali Kanwal*

*Aser Ahmad Ibrahim Ahmad*

*Jean Wilbert Folsy*

*Zayed Hassan*

**CS435**

**Math Problem 1**

To know if a function is increasing/eventually decreasing

* Get its derivative, .
* Get the zero of the derivative .
* Test before and after .

Test before and after

From (1), (2), and (3):

Test before and after :

From (1), (2), and (3):

We note that for any value of , will be positive. This means that .

**Math Problem 2**

To know how a function grows with respect to another function , we divide them:

* If then asymptotically grows no faster than .
* If then asymptotically grows no faster than .
* If then and asymptotically grow with the same rate.

**Problem 1:** GCD Algorithm

private static int gcd(int m, int n) {

int result = 1;

for (int i = 1; i <= m && i <= n; i++) {

if (m % i == 0 && n % i == 0)

result = i;

}

return result;

}

**Problem 2:** Subset sum problem – Brute Force Solution

private static int[] SubsetSum(int[] S, int k) {

Arrays.sort(S);

int[] T = new int[0]; //this array will hold the required subset

int size = 0; //the final size of the required subset

for (int i = S.length - 1; i >= 0; i--) { //scan the array from the end

int temp = k;

T = new int[S.length];

int TSize = 0;

for (int j = i; j >= 0; j--) {

if (temp - S[j] >= 0) {

temp -= S[j];

T[TSize] = S[j];

TSize++;

}

}

//if we found the sum. We get the found subset size

if (temp == 0) {

size = TSize;

break;

}

}

//copy the found subset to a new array with exactly the found subset size

int[] result = new int[size];

for(int i=0 ; i<size ; i++){

result[i] = T[i];

}

return result;

}

**Problem 3:** Greedy strategies

private static int[] GreedySubsetSum(int[] S, int k) {

Arrays.sort(S);

int[] T = new int[0]; //initialize the subset

int size = 0; //initialize the subset size

for (int i = 0; i < S.length; i++) {

int temp = k;

T = new int[S.length];

int TSize = 0;

for (int j = i; j < S.length; j++) {

if (temp - S[j] >= 0) {

temp -= S[j];

T[TSize] = S[j];

TSize++;

}

}

if (temp == 0) {

size = TSize;

break;

}

}

//copy the found subset to a new array with the exact size

int[] result = new int[size];

System.arraycopy(T,0,result,0,size);

return result;

}

**Problem 4**

The solution will always be true because if we remove last element from *S* (which is also an element of *T*)

then it will be equal to the subsetSum of K', where K' is *K – last element*. Also, we remove the element from T.

For example, S = {4, 5, 10, 12};

K = 22;

T = {10, 12};

Now, S' = {4, 5, 10};

K' = K – 12 = 10

T' = T – {12} = {10}

Here K' is the sum of T' subset.