CS101
Data Structures and Algorithms

Lecture 06

Complexity vs Loops

Simple O(n) algorithms -1

- Description: Compute the length of the given list with sentinel -1
- Note; Sentinel is the final / last input value.

```
for(i=0; nlist[i] != -1; i++):
Report i as list length
```

Complexity is O(n) as we iterate the entire input list

Simple O(n) algorithms -2

- Description: Compute sum of the list of n numbers, say nlist

```
sum = 0
for(i=0; i<n; i++):
    sum += nlist[i]
Report sum
```

Complexity is O(n) as there is only one loop that runs n times.

Simple O(n) algorithms -2

- Description: Compute sum of the list of n numbers, say nlist
- What if we dont wish to use the length of the list?

```
sum = 0
foreach item in nlist:
    sum += item
Report sum
```

Simple O(n) algorithms -3

- Description: Compute the mean of input numbers read interactively that treats -1 as the sentinel (end of input)

```
sum = 0
for(i=0; ;i++) : #No condition used
    x = read input integer
    if (x == -1):
        break  # Leave the loop
    sum += x
Mean = sum/i
Report Mean
```

Simple O(n) algorithms -4

- Description: Compute the mode of input values read interactively that treats -1 as the sentinel (end of input). Note input value is in the range [0, 20]

```
for(i=0;i<21;freq[i]=0,i++):

for(i=0; x[i]!=-1; i++):
    freq[x[i]]=freq[x[i]]+1

for(max=0,i=0; x[i]!=-1; i++):
  if(max < freq[x[i]]:
      max=freq[x[i]];
```

Simple O(n) algorithms -5

- Description: Compute the mode in a list X of arbitrary input values
- First sort the input list X, and then do the following
- Keep count of successive values that are same. If the value changes then update the max-value-count

```
X = sort(X)

v = X[0], vc = 0, mvc = 0

foreach item in X:
    if (v == item):
        vc++
    else
        if( vc > mvc ):
            mvc = vc
            mode = v
        vc=1, v=item

Report mode
```

Simple O(n) algorithms -6

- Description: Cumulative Sum or Prefix Sum of a given list X
- Output: A list of prefix sums

```
Y = [] (or Y = list())

S = 0

foreach item in X:
        s += item
        Y.append(s)
```

Simple $O(n^2)$ algorithms -1

- Description: Determine whether two input lists are disjoint
- This algorithm uses flag technique
- When two elements have same value Then we set a "flag" variable
- As there two nested loops the complexity is $O(n^2)$

```
found = 0 # or FALSE

for a in A:
    for b in B:
        if (a == b):
            found = 1
            break

    if(found==1)
        break

if( found == 1)
    print "A and B not disjoint)
else
    print "A,B are disjoint"
```