# Lab 2a

1) A)

```
sum ← 0                                  1
for i ← 0 to n-1 do                      1 + (n) + 1
    sum ← sum + 1                        2n
    { incrementation i }                 2n
```

5n+3

: n times.

5)

```
sum ← 0                                  1
for i ← 0 to n-1 do                      n - 1
    for j ← 0 to n-1 to                  (n+2) n
        sum ← sum + 1                    2n * n
        { incrementation j }             2n * n
{ incrementation i }                     2n
```

total: $1 + n + 2 + (n^2 + 2n) + 2n^2 + 2n^2 + 2n$

$\Rightarrow 1 + n + 2 + 5n^2 + 2n + 2n$

$\Rightarrow 5n^2 + 5n + 3$

2) first for loop gives $O(n)$

second for loop gives $O(n^2)$

$\Rightarrow O(n^2)$

3) a Algorithm merge(A, B)

input : sorted array size x
~~output~~ sorted array size y
output : Array C with size n which merge A and B

```
i ← 0
j ← 0
n ← x + y
C ← new int[n]
index ← 0
while i<x and j<y do
    if A[i] <= B[i] then
        C[index] = A[i]
        {increase index}
        {increment i}

    [else]
        C[index] = B[j]
        {increment index}
        {increment j}

while i<x do
    C[index] = A[i]
    {increment i}
    {increment index}
```

while j < y do

c [index] = B[j]

{increment i}

{increment j}

return c.

3)B) while loop $\simeq O(n)$

while loop $+ O(n-x)$

while loop $+ O(n-x)$

$O(n)$

$\simeq O(n)$

4)A) Algorithm remove_dups (L)

input List L

output List M containing the distinct elements of L

M ← new List

for i ← 0 to L.size (j-1) do

if not M contains (L[i]) then M.add (L[i])

return M

here

for loop → ~~repeat~~ n operations

$\Rightarrow O(n^2)$

4)3 Algorithm removeDups

input a List L

output a list M containing the distinct elements of L

M ← new hashmap

for i ← 0 to L.size() - 1 do

if not m.contains [L[i]) then

M.put [L[i], 0]

return M.

Ans: Algorithm B of removeDups include a hashmap data structure

→ contains → O(1) operations

→ for loop → O(n)

contains O(1)

O(n) * O(1) => O(n)

→ this algorithm is more efficient (O(n)) than the last one.

4) c

to prove algorithm B is correct

loop invariant, for $0 \leq i < n-1$

I(i) : M contains the distinct elements contained [L[0] .. L[i]]

base case:

I(0) : M contains the distinct elements contains L[0].

induction step

we assume I(i) : M contains distinct elements of L[0] to L[i]

show that

I(i+1) : M contains elemen [L[0] ... L[i+1]]