# Data Structure Assignment #1

# Performance Analysis and Measurement

# (Submit solution to my e-mail: utilForever@gmail.com)

1. Permutations 함수의 시간 복잡도는?

void Permutations(char\* a, const int k, const int m)

{

// Generate all the permutations of a[k], ..., a[m].

if (k == m) // Output permutation

{

for (int i = 0; i <= m; i++)

*cout* << a[i] << " ";

*cout* << *endl*;

}

else // a[k:m] has more than one permutation. Generate these recursively.

{

for (int i = k; i <= m; i++)

{

*swap*(a[k], a[i]);

Permutations(a, k + 1, m);

*swap*(a[k], a[i]);

}

}

}

1. Magic 함수의 시간 복잡도는?  
   (여기서 말하는 Magic Square란 모든 행, 열, 대각선의 합이 같은 행렬을 말함)  
   (H. Coxeter는 이 홀수일 때 Magic Square를 만드는 간단한 방법을 제시함)

void Magic(const int n)

{ // Create a magic square of size n, n is odd.

const int MaxSize = 51; // Maximum square size

int square[MaxSize][MaxSize], k, l;

// Check correctness of n

if ((n > MaxSize) || (n < 1))

throw "Error! n out of range";

else if (!(n % 2)) throw "Error! n is even";

// n is odd. Coxeter's rule can be used

for (int i = 0; i < n; i++) // Initialize square to 0

*fill*(square[i], square[i] + n, 0); // STL algorithm

square[0][(n - 1) / 2] = 1; // Middle of first row

// i and j are current position

int key = 2, i = 0, j = (n - 1) / 2;

while (key <= n \* n)

{

// Move up and left

if (i - 1 < 0) k = n - 1;

else k = i = 1;

if (j - 1 < 0) l = n - 1;

else l = j - 1;

if (square[k][l]) i = (i + 1) % n; // Square occupied, mvoe down

else

{ // square[k][l] is unoccupied

i = k;

j = l;

}

square[i][j] = key;

key++;

} // End of while

// Output the magic square

*cout* << "Magic square of size " << n << *endl*;

for (i = 0; i < n; i++)

{

*copy*(square[i], square[i] + n, *ostream\_iterator*<int>(*cout*, " "));

*cout* << *endl*;

}

}

1. SelectionSort 함수의 시간 복잡도는?

void SelectionSort(int\* a, const int n)

{ // Sort the n integers a[0] to a[n-1] into nondecreasing order.

for (int i = 0; i < n; i++)

{

int j = i;

// Find smallest integer in a[i] to a[n - 1]

for (int k = i + 1; k < n; k++)

if (a[k] < a[j])

j = k;

*swap*(a[i], a[j]);

}

}

1. 다음 순환식의 시간 복잡도를 구하고, Master Theorem을 통해 맞는지 확인해 보라.

|  |
| --- |
| - The Form :  - The Master Theorem : ( is any constant)  1. If is , then is .  2. If is , then is .  3. If is , then is , provided for some . |

1. 복소수 행렬 는 와 가 실수 값으로 이루어진 행렬이라고 할 때, 행렬 쌍 로 표현할 수 있다. 두 복소수 행렬 와 의 곱을 계산하는 프로그램을 작성하라. 여기서 다. 모든 행렬이 일 때 덧셈과 곱셈의 횟수는 얼마인가?