알고리즘분석 HW3 학습내용

2017104017 컴퓨터공학과 이준용

1번 문제)

1번 문제에서 QuickSort를 재귀적으로 호출하는데 그 중 partition 과정은 다음과 같이 이루어진다.

1. pivotItem을 설정한다. (어떤 Item을 골라도 상관없으나 편의상 첫번째 item으로 설정함)
2. j=low로 설정한다.
3. i = low +1부터 시작하여 배열의 끝까지 반복을 돈다.
4. 만약 배열의 i번째 item이 pivotitem보다 작으면 j = j+1을 하고 j번째 item과 i번째 item을 swap한다.
5. 배열 끝까지 돌면 j번째 아이템과 pivotItem(s[low]로 설정)을 교환한다.

quicksort 알고리즘의 시간복잡도를 계산할 때 단위연산을 s[i]와 pivotItem의 비교라고 가정한다면 최악의 경우는 비내림차순으로 정렬되어 있을 경우이고, 항목의 수를 n이라 가정하면

$T(n) = T(n-1) + n -1$

이 된다. 이것은 재귀적으로 풀면

$T(n) = n(n-1)/2$

이 된다.

2번 문제)

이차원 행렬의 곱셈은 단순히 계산하면 8번의 곱셈과 4번의 덧셈이 필요하고, n차원 행렬의 곱셈은 시간복잡도가 n^3 이 된다. N이 커진다면 시간복잡도가 어마어마하게 커지므로 threshold보다 n이 클 경우 사용하는 알고리즘이 Strassen알고리즘이다. Strassen 알고리즘은 C =AXB 라는 행렬이 있을 때 A와 B를 A11,A12,A21,A22와 B11,B12,B21,B22의 작은 행렬로 나누어

M1 = (A11+A22) X (B11+B22)

M2 = (A21+A22) X B11

M3 = A11 X (B12-B22)

M4 = A22 X (B21-B11)

M5 = (A11+A12) X B22

M6 = (A21-A11) X (B11 + B12)

M7 = (A12-A22) X (B21+B22)

를 계산하고,

C = M1+M4-M5+M7 M3+M5

M2+M4 M1+M3-M2+M6 방식으로 계산하는 방법으로

총 곱셈의 횟수가 7번 덧셈뺄셈의 수가 18번 필요하므로 위의 방법보다 n이 큰수에 대해서 시간 복잡도가 감소한 것을 알 수 있다.