DAA Algorithms

1 QuickSort $O(n \log n)$

```
Quick_sort (A,p,r)
1  if p<r
2      q ← partition (A,p,r)
3      Quick_sort (A,p,q-1)
4      Quick_sort (A,q+1,r)
```

```
Partition (A,p,r)
1.  x ← A[r].
2.  i ← p-1.
3      for j ← p to r-1
4.         do if A[j] < x
5              then i ← i+1.
6              exchange A[i] <-> A[j].
7              exchange A(i+1) <-> x
8.  return i+1.
```

2. Mergesort — O(nlogn)

```
Merge_sort (A,p,r)
1. if p<r
2. then q ← ⌊(p+r)/2⌋
3.      Mergesort (A,p,q)
4.      Mergesort (A,q+1,r)
5.      Merge(A,p,q,r)
```

```
Merge(A,p,q,r)
1. n1 ← q-p+1
2. n2 ← r-q
3.      for i ← 1 to n1
4.              do L(i) ← A(p+i-1)
5.      for j ← 1 to n2
6.              do R(j) ← A(q+j)
7.      L(n1+1) ← ∞
8.      R(n2+1) ← ∞
```

3. Strassens matrix. $O(n^{\log 7})$

$$\begin{bmatrix} A11 & A12 \\ A21 & A22 \end{bmatrix} \times \begin{bmatrix} B11 & B12 \\ B21 & B22 \end{bmatrix} = \begin{bmatrix} C11 & C12 \\ C21 & C22 \end{bmatrix}$$

$C11 = P + S - T + V$

$C12 = R + T$

$C21 = Q + S$

$C22 = P + R - Q + U$

$P = (A11 + A22)(B11 + B22)$

$Q = (A21 + A22) B11$

$R = A11 (B12 - B22)$

$S = A22 (B21 - B11)$

$T = (A11 + A12) B22$

$U = (A21 - A11)(B11 + B12)$

$V = (A12 - A22)(B21 + B22)$

```
procedure strassen(n, A, B, C)
begin
  if n = 2
    c11 = a11·b11 + a12·b21
    c12 = a11 b12 + a12·b22
    c21 = a21·b11 + a22·b21
    c22 = a21·b12 + a22 b22
  else
  Partition A into 4 submatrices
    A11, A12, A21, A22;
  Partition B into 4 submatrices;
    B11, B12, B21, B22;

  call strassen.
    (n/2, A11+A22, B11+B22; P)

  call strassen.
    (n/2, A11+B12-B22, R);

  call strassen.
    (n/2, A22, B21-B11, S);

  call strassen;
    (n/2, A11+A12, B22);

  call strassen(n/2, A21-A11, B11+B12, U)

  call strassen(n/2, A12-A22, B21+B22,
                V);
  C11 = P+S-T+V; C12 = R+T;
  C21 = Q+S; C22 = P+R-Q+U;
  end;
```

4. Maximin O(n)

```
procedure RMaxMin (i, j, fmax, fmin);
begin.
    case :
    i = j       fmax ← fmin ← A(i);
    i = j-1     if A(i) < A(j)
                then [ fmax ← A(j);
                       fmin ← A(i);
                else [ fmax ← A(i);
                       fmin ← A(j);.
    else:       mid ← [ (i+j)/2 ];

                call RMaxMin( i, mid, gmax, gmin);
                call RMaxMin( mid+1, j, hmax, hmin);
                fmax ← MAX(gmax, hmax);
                fmin ← MIN (gmin, hmin);
    end.
end.
```

3. Fractional knapsack Greedy

1. sort n objects from large to small based on ratios vi/wi.
2. We assume arrays w(1...n) and v(1...n) store respective weights and values after sorting.
3. initialize array x[1...n] to 0
4. weight = 0; i = 1.
   value = 0
5. while(i ≤ n and weight < W) do.
6. if weight + w(i) ≤ W then.
7. x[i] = 1
8. else
   x[i] = (W - weight) / w(i)
9. weight = weight + x(i) * w(i)
10. i++;

6. Kruskal's algorithm. $O(E \log V)$

MST_KRUSKAL(u, w)

1. A ← ∅ vertex
2. for each v ∈ V(u)
3. do make set v
4. sort the edge E by weight (↑ order).
5. for each edge (u, v) ∈ E taken in increasing order.
6. do if findset(u) ≠ findset(v)
7. then A ← A ∪ {u, v}
8. union(u, v).
9. findset(u), findset(v).
10. return A

end.

7. Prim's Algorithm. $O(E + V \cdot \log V)$

1. for each vertex $v \in V(u)$
2. do key[v] $\leftarrow \infty$
3. $\pi(v) \leftarrow$ Nil
4. key(r) $\leftarrow 0$.
5. $Q \leftarrow V(u)$.
6. while $Q \neq \emptyset$.
7. do $u \leftarrow$ EXTRACT-MIN(Q).
8. for each vertex $v \in Adj(u)$
9. do if $v \in Q$ and $w(u,v) <$ key(v).
10. change key(v) $\leftarrow w(u,v)$.
11. $\pi(v) \leftarrow u$.

8. N-Queens $O(n!)$

```
Algorithm NQueens(k,n)
{ for i:=1 to n do.
  {
    if place(k,i) then.
      {
        x(k)=i;
        if (k=n) then write (x[1:n]);
        else NQueens(k+1,n);
  } }.
```

```
Algorithm Place(k,i)
{
  for j=1 to k-1 do
    if ((x(j)=i)
        or (Abs(x(j)-i)=Abs(j-k)))
    then return false;
  return true;
}.
```