Algorithm Notes

Network Flow Algorithms

FF - for any flow, any s-t cut(L,R): size(f) <= c(L,R)

- Edmonds-karp - shortest path first
- Capacity-scaling - augment path of largest capacity first, for each iteration, filter out your residual graph by the edges that have capacity > D; halve D each iteration
    - Property 1: While loop 1 is executed 1 + log2 Cmax times
    - Property 3: For any D, #iterations of loop 2 in the D-scaling phase <= 2|E|
    - Property 2: At the end of a D-scaling phase, size(max flow) <= size(current flow) + D|E|

Farthest First K-Means

- Property 1. Solution value of FF-traversal = r
- Property 2. There are at least k+1 points S s.t each pair has distance >= r, where S = C U {q}.
- Property 3. The Optimal solution must assign at least two points x, y in S to the same center c
- Property 4. Thus, Opt. solution has value >= r/2

Greedy Set Coverage - find smallest subset C* of C (collection of sets) that covers all of U

- n(t) = #uncovered elements after step t in greedy
- Property 1: There is some S that covers at least n(t)/k of the uncovered elements
- Property 2: n(t+1) <= n(t)(1 - 1/k)
- Property 3: n(T) <= n(1 - 1/k)T < 1, when T = k ln n

MST

- Properties
    - Property 1. A lightest edge in any cut always belongs to an MST
    - Property 2. The heaviest edge in a cycle never belongs to an MST
- Prims - adds the lightest edge between connected and non-connected sets each iter.
    - O(mlogn) where m = |E|
    - Heap: add: O(logn), delete: O(logn), report min O(1)
- Kruskal's - in order of edge weight, if two endpoints not connected, connect them
    - O(mlogn) where m = |E|
    - Property 1: If x is not a root, then rank[p[x]] > rank[x]
    - Property 2: For root x, if rank[x] = k, then subtree at x has size >= 2k
    - Property 3: There are at most n/2k nodes of rank k
    - Property 1: Total time for m find operations = O((m+n) log*n)
    - Property 2: Time for each union operation = O(1) + Time(find)

Independent set - subset of vertices that share no edges between them

A connected, undirected and acyclic graph is called a tree

- Property 1. A tree on n nodes has exactly n - 1 edges
- Property 2. Any connected, undirected graph on n nodes and n - 1 edges is a tree

Binary Tree

- The number of nodes n in a perfect binary tree can be found using this formula: $n = 2$ ^$(h + 1) - 1$ where h is the depth of the tree.
- The number of nodes n in a complete binary tree is at least $n = 2h$ and at most $n = 2$ ^$(h + 1) + 1 -$

1 where h is the depth of the tree.

- The number of leaf nodes L in a perfect binary tree can be found using this formula: $L = 2^h$ where h is the depth of the tree.
- The number of nodes n in a perfect binary tree can also be found using this formula: $n = 2L - 1$ where L is the number of leaf nodes in the tree.
- The number of null links (absent children of nodes) in a complete binary tree of n nodes is $(n+1)$.
- The number $n-L$ of internal nodes (non-leaf nodes) in a Complete Binary Tree of n nodes is $\lfloor n / 2 \rfloor$.
- For any non-empty binary tree with $n0$ leaf nodes and $n2$ nodes of degree 2, $n0 = n2 + 1$

Divide and Conquer

Matrix Multiplication

- Algorithm 1: 1) Compute AE, BG, CE, DG, AF, BH, CF, DH; 2) Compute AE + BG, CE + DG, AF + BH, CF + DH
- Operations: 8 n/2 x n/2 matrix multiplications, 4 additions
- Recurrence: T(n) = 8 T(n/2) + O(n2) T(n) = O(n3)

Closest Pair

Find-Closest-Pair(Qx,Qy):
1. Find the median x coordinate lx
2. L = Points to the left of lx
3. R = Rest of the points
4. (a1, b1, t1) = Find-closest-pair(Lx, Ly)
5. (a2, b2, t2) = Find-closest-pair(Rx, Ry)
6. t = min(t1, t2)
7. Let Sy = points within distance t of lx sorted by y
8. For each p in Sy
Find distance from p to next 15 points in Sy
Let (a, b) be the closest such pair
9. Report the closest pair out of:
(a, b), (a1, b1), (a2, b2)

Int Mult

- xLyL 2^n + (xR yL + xLyR) 2^(n/2) + xRyR

String reconstruction (sequence of words)

- S(k) = True iff there is j < k s.t. S(j) is True, and x[j+1..k] is a valid word

LCS

- S(i,j)=S(i-1,j-1) + 1, if x[i] == y[j] ; = max(S(i-1,j), S(i,j-1)), ow

Edit Distance

- E(i,j) = min(E(i-1,j) + 1, E(i, j-1) + 1,E(i-1,j-1) + diff(x[i], y[j]))
    - diff(a,b) = 0 if a==b; = 1 ow

Subset Sum - Given a list of positive integers a[1..n] and an integer t, is there some subset of a that sums to exactly t?

- If a[i] s, then S(i,s) = S(i - 1, s - a[i]) OR S(i - 1, s)
- Else: S(i, s) = S(i - 1, s)
- Reconstructing the subset
    - If S(i, s) = True, and S(i-1,s-a[i]) = True then D(i, s) = (i - 1, s - a[i])
    - OW: D(i, s) = (i - 1, s)