# Final Exam 'Cheat Sheet'

**David Walter**

**Speeding up Djikstra**: single-pair shortest path: terminate once extract t from priority queue.
Speed up for highly branched graphs:
**Bi-directional** Search w/ Djikstra:
Estimate diameter of graph to be log_b(n), where b is the branching factor, and the last layer out dominates the num of nodes.
How: run forward djikstra from s, and backward djikstra from t, and stop once the two searches meet.
Stopping condition: NOT (just when the same edges are met or when the same vertices are extracted from priority queue.).

**DP: Longest Path**: O(V+E). topologically sort and then work backwards, finding longest path from predecessor vertices and recursively find the answer to each sub problem.

**DP: Edit Distance**: O(mn)

$$S[i][j] = \begin{cases} S[i-1][j-1] & \text{if } A[i] = B[j] \\ 1 + \min\{S[i][j-1], & \\ \quad S[i-1][j], S[i-1][j-1]\} & \text{otherwise.} \end{cases}$$

**DP: LCS: Longest common subsequence**: O(mn), where n=len(A), m=len(B), O(mn) memory

$$S[i][j] = \begin{cases} S[i-1][j-1]+1 & \text{if } A[i] = B[j] \\ \max\{S[i][j-1], S[i-1][j]\} & \text{otherwise} \end{cases}$$

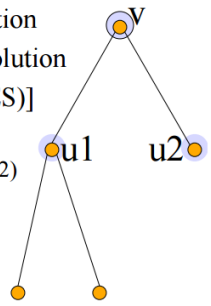
Three substrings:O(mnp)

$$S[i][j][k] = \begin{cases} 1 + S[i-1][j-1][k-1] & \text{if } A[i] = B[j] = C[k] \\ \max\{S[i][j][k-1], S[i][j-1][k], S[i-1][j][k]\} & \text{otherwise.} \end{cases}$$

## Vertex cover: algorithm

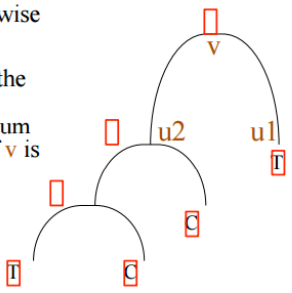
- Sub-problems: let cost(v,b) be the min-cost solution of the sub-tree rooted at v, assuming v's status is b∈{YES,NO}:
  - Cost(v,YES): assuming v is in the solution
  - Cost(v,NO): assuming v is not in the solution
- solution=min[Cost(root,NO),Cost(root,YES)]
- Recurrence for cost(v,b) ?
  cost(v,YES)= 1+$\min_{b1}$ cost(u1,b1)+ $\min_{b2}$cost(u2,b2)
  cost(v,NO)= cost(u1,YES)+cost(u2,YES)
- Base case v=leaf:
  cost(v,YES)=1
  cost(v,NO)=0

– Start from the leaves
– Proceed bottom up the tree
Total time ? O(n)




## Parsimony: dynamic program

- Define letter distance as follows
  D(a,b)=0 if a=b and =1 otherwise
- Sub-problem: For any node v of the tree and label L in {A, C, G, T}, define **cost(v,L)** to be the minimum cost for the subtree rooted at v, if v is labeled L
- solution=$\min_L$ cost(root,L)
- Recurrence for cost(v,L) ?
  cost(v,L)=$\min_{L1, L2}$ [ D(L,L1) + cost(u1,L1) ] + [ D(L,L2) + cost(u2,L2) ]
- Base case: if v is a leaf
  cost(v,L)= ∞*D(L,leaf_label(v))




Running time?
O(n k) * O(k) = O(nk²)
where k is the alphabet size

## Newton's Method/ Numerics:

$f(x) \approx f(x_0) + f'(x_0)(x - x_0).$

**Division** $f(x) = \frac{1}{x} - \frac{b}{R}$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{\left(\frac{1}{x_i} - \frac{b}{R}\right)}{-1/{x_i}^2} = x_i + {x_i}^2\left(\frac{1}{x_i} - \frac{b}{R}\right) = 2x_i - \frac{b{x_i}^2}{R}$$

**Cube Roots**

$$f(x) = x^3 - a$$
$$f'(x) = 3x^2$$
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} = x_i - \frac{x_i^3 - a}{3x_i^2} = \frac{2x_i}{3} + \frac{a}{3x_i^2}$$

$$x_{i+1} := x_i - \frac{x_i^2 - a}{2x_i} = x_i - \frac{1}{2}x_i + \frac{1}{2}\frac{a}{x_i} = \frac{1}{2}\left(x_i + \frac{a}{x_i}\right)$$

$\Theta(d^\alpha)$

$T(n) = \Theta(n^{\log_2 3})$

```
procedure karatsuba(num1, num2)
  if (num1 < 10) or (num2 < 10)
    return num1*num2
  /* calculates the size of the numbers */
  m = max(size_base10(num1), size_base10(num2))
  m2 = m/2
  /* split the digit sequences about the middle */
  high1, low1 = split_at(num1, m2)
  high2, low2 = split_at(num2, m2)
  /* 3 calls made to numbers approximately half the size */
  z0 = karatsuba(low1,low2)
  z1 = karatsuba((low1+high1),(low2+high2))
  z2 = karatsuba(high1,high2)
  return (z2*10^(2*m2))+((z1-z2-z0)*10^(m2))+(z0)
```

INTEGER-ADD$(X, Y, d, Z)$

```
1  carry := 0
2  for i := 0 to d − 1
3       do r := X[i] + Y[i] + carry
4          Z[i] := r mod Y
5          carry := ⌊r/Y⌋
6  if carry > 0
7     then ARRAY-APPEND(Z, carry)
```

This algorithm runs in time $\Theta(d)$. N

Sp14 exam makeup:

-master thm/recurrences checking-5pts

- Qs about sorting algorithms – 5pts

-Qs about arbitrary BSTs -5pts

-Qs about hashing -5pts

-Qs about Newton's Method -5pts

-Qs about DFS -5pts

-Qs about topological sorting -5pts

-Qs about DP & subproblems -5pts

---

-describe algo for priority queue -10pts

-jug problem and BFS -10pts

-prove why cant fix neg weights -10pts

-bi-directional djikstra – 15pts

-DP problem – 15pts

-DP problem – 15pts

---

-hashing pseudo code -20pts

- using bellman-ford -20pts

-scheduling using DP -20pts