Chimzim Ogbondah

Com S 311

16 April 2021

Homework 4

1. Matrix
   1. public int FINDMAXPATH**(int w, int m, int n)**
   2. i = 1, j= 1, finished = 0, sumWeight = 0, maxWeight = -∞;
   3. String direction = “”;
   4. **while** (finished != 1)
   5. **if** (i ≥ m && j ≥ n) : finished = 1;
   6. **else :**
   7. **else if** (i ≥ 1 && i+1 ≤ m)
   8. int t = i+1
   9. temp\_weight = MAX(maxWeight, weight[t, 1])
   10. if(maxWeight == temp\_weight): direction = “vertical”
   11. else if( j ≥ 1 && j+1 ≤ n):
   12. int t = j+1
   13. get weight from w[j, t-1] 🡪w[j, t]
   14. temp\_weight = MAX(maxWeight, weight[1, j])
   15. if(maxWeight == tempWeight): direction = “horizontal”
   16. else if(i ≥ 1 && i+1 ≤ m )&& (j ≥ 1 && j+1 ≤ n):
   17. int a = i+1
   18. int b = j+1
   19. get weight from w[a-1, b-1] 🡪 w[a,b]
   20. temp\_wieght = MAX(maxWeight, weight[a,b])
   21. if(temp\_weight == maxWeight) : direction = “diagonal”
   22. ---------------
   23. If(direction == “vertical”) : i++
   24. Else if(direction == “vertical”): j++
   25. Else : i++ j++
   26. sumWeight = sumWeight + maxWeight
   27. maxWeight = -∞;
   28. end Whie
   29. return sumWeight /\*max weight path number\*/
2. Prove P is closed under the STAR operation by dynamic programming.
   1. Let A be a Language in P and Let B be a Turning Machine deciding A in polynomial time. The Following Decides A\*:
   2. ST = on input x = x1x2x3….xn
   3. If x == empty **Accept**
   4. Init Arr[I, j] = 0 for 1 ≤ i ≤ j ≤ n
   5. For i = 1 to n: Set Arr[i, i] = 1 if xi is in A
   6. For c = 2 to n:
   7. For I = 1 to n-c+1:
   8. J = i+c-1
   9. If xi…xj is in A set Arr[i, i] = 1
   10. For k = i to j-1:
   11. If Arr[i, k] = 1 && Arr[k, j] = 1 set Arr[i, j] = 1
   12. IF Arr[1, n] = 1 **Accept** otherwise **Reject**
3. TRIANGLE PROBLEM
   1. Let G = (V, E) be a graph with a set of vertices V and a set of edges E. Enumerating all tripes (x, y, z) with vertices x, y, z ∈ V where x < y < z.
   2. Check if all 3 edges (x,y) (y, z) and (x, z) exist inside E. This will take O(||) time. Checking if all edges belong to E will take O(|E|) time which will give it an over all runtime of O(|). This is polynomial in length to the input <G>. So, 3-ANGLE ∈ P
4. If P=NP, then we can factor integers in polynomial time
   1. PROOF:
   2. Let L be a language = { (n, t) | n and t are non-zero and negative factors where n has a factor f which satisfies t ≤ f < n }
   3. Show that L ∈ NP by using a Non-Deterministic Turning Machine A which decides L by guessing (choosing) and integer f verifying t ≤ f < n and f divides n.
   4. By invoking A with inputs <n, 2> we will check if n is PRIME or not. If n is not PRIME we use A to discover the PRIME factorization of n:
   5. Let F(largest) be the largest factor lying from (1, n). A accepts <n, f> for al f ∈ [2, F(largest)]. A rejects <n, f> for all f ∈ [F(largest)+1, n].
   6. F(largest) can be found by iterating through f 1->n. Knowing F(largest) we can compute p = n/F(largest). Then finding the largest factor n/p using the above procedure for finding F(largest)
   7. For any n we have O(log n) prime factors and each prime factorization has O(n) invocations to A for its discovery. A is invoked O(n Log n) times. Assuming P = NP the whole procedure is polynomial