Analysis and Design of Algorithms

Algorithms
CS3230

Tutorial

Week 3

Question 1

Dijkstra(G, s)

1. *For all $u \in V \backslash \{s\}$, $d(u)=\infty$;*
2. *$d(s)=0$; $R=\{\}$;*
3. *While $R \neq V$*
4. *pick $u \notin R$ with the smallest $d(u)$*
5. *$R = R \cup \{u\}$*
6. *for all neighbor v of u,*
7. *$d(v) = \min\{ d(v), d(u)+w(u,v))\}$*

G=(V,E) is an undirected graph.
s is the start node
Assume all edges in G are of positive weights.

What is the invariant for the while loop?

Can you show that this algorithm correctly compute the shortest distance from s to all nodes?

Answer

Let $\delta(a \rightarrow b)$ be the shortest distance from a to b in G.
Let $\delta(X,u)$ be $\min\{\delta(s \rightarrow v)+w(v,u) \mid v \in X\}$.

Invariant:
1. For all $u \in R$, $d(u)=\delta(s \rightarrow u)$. (i.e d(u) is the shortest distance from s to u.)
2. For every neighbor u of R, $d(u)=\delta(R,u)$

Answer

Initialization: Before the first iteration of the while loop, R={}. The invariant is true.

Answer

Maintenance: Note that the size of R is increasing. For every iteration, a new node u is inserted into R. Let $R' = R \cup \{u\}$.

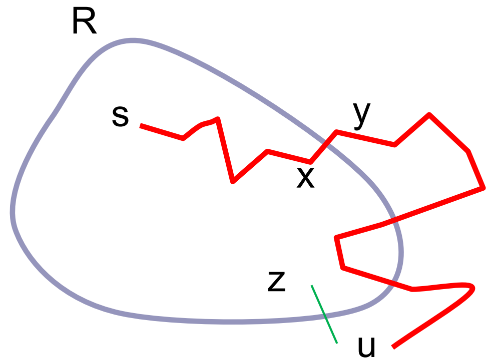
To keep the invariant, we need to show that

1. $d(u) = \delta(s \rightarrow u)$ = the shortest distance from s to u.
2. For every neighbor v of u, $d(v) = \delta(R', v)$

For (1), by contradiction, assume d(u) is not the shortest distance.

There exists a shortest path Q from s to u which is shorter than d(u).

Answer



Since $s \in R$, the path Q moves from R to $V \backslash R$. Let (x,y) be the first edge leaving R in Q.

By the definition of Q, the weight of Q is $\delta(s \rightarrow x)+w(x,y)+\delta(y \rightarrow u)<d(u)$.

The algorithm maintains $d(y)= \delta(R,u)=\min\{ \delta(s \rightarrow r)+w(r,y) \mid r \in R\}$.
So, we have $d(y) \leq \delta(s \rightarrow x)+w(x,y)$.
Hence, $d(y) \leq d(y)+\delta(y \rightarrow u) \leq \delta(s \rightarrow x)+w(x,y) + \delta(y \rightarrow u) < d(u)$.

The algorithm selects u instead of y in step 4 implies that $d(u) \leq d(y)$.
Contradiction.

Answer

For (2), we need to show that, for every neighbor v of u, $d(v)=\delta(R',v)$.

Before steps 6-7, the invariant implies $d(v) = \delta(R,v)$.
Note that $R'=R\cup\{u\}$.
So, we need to update d(v) to min $\{\delta(R,v), d(u)+w(u,v)\}$.

This is done in Step 7. It sets d(v) = min{ d(v), d(u)+w(u,v))}.

Answer

Termination: After the last iteration of the while loop, R includes all vertices in G. So, d(u) is the shortest distance to s for every node u in G.

Question 2

Which of the following statement is false?

- $\log(f(n)g(n)) = O(\max(\log f(n), \log g(n)))$
- $2^{n+1} = O(2^n)$
- $(n+1)! = O(n!)$
- $(n+a)^b = \Theta(n^b)$

Answer

Answer: (3)

$$(n+1)! = (n+1) \times n!$$

is not $O(n!)$ as

$$\frac{(n+1)!}{n!} = n + 1 > C$$

for any constant C

Answer

Other answers are correct:

1) $\log f(n)g(n) = \log f(n) + \log g(n)$
$= O(\max\{\log f(n), \log g(n)\}$

2) $2^{n+1} = 2 \times 2^n = O(2^n)$

4) $(n+a)^b = \Theta(n^b)$
as $(n+a)^b$ is a polynomial of degree *b*

Question 3

Use the master method to give tight asymptotic bound for $T(n) = 2T(n/4) + n$.

- $\Theta(\log n)$
- $\Theta(\sqrt{n})$
- $\Theta(n)$
- $\Theta(\sqrt{n}\log n)$

Answer

Answer: (3)

$T(n) = 2T(n/4) + n$

Master Theorem case 3

and $$f(n) = n = \Omega(n^{\log_b a+\epsilon}) = \Omega(n^{1/2+\epsilon})$$

$$af(n/b) = 2f(n/4) = 2n/4 = n/2 \leq f(n) = n$$

Hence $T(n) = \Theta(n)$

Question 4

Use telescoping to give a tight asymptotic bound for $T(n) = 4T(n/4) + n/\lg n$.

1. $\Theta(n)$
2. $\Theta(n \lg n)$
3. $\Theta(n \lg \lg n)$
4. $\Theta(n^2)$

Answer

Answer: (3)

$$T(n) = 4T(n/4) + n/\lg n \rightarrow \frac{T(n)}{n} = \frac{T(\frac{n}{4})}{n/4} + \frac{1}{\lg n}$$

Telescoping!

$\lg_4 n$

$$\frac{T(n)}{n} = \cancel{\frac{T(n/4)}{n/4}} + \frac{1}{\lg n}$$

$$\cancel{\frac{T(n/4)}{n/4}} = \cancel{\frac{T(n/4^2)}{n/4^2}} + \frac{1}{\lg(n/4)}$$

$$\cancel{\frac{T(n/4^2)}{n/4^2}} = \cancel{\frac{T(n/4^3)}{n/4^3}} + \frac{1}{\lg(n/4^2)}$$

$$\vdots$$

$$\cancel{\frac{T(4)}{4}} = \frac{T(1)}{1} + \frac{1}{\lg 4}$$

$$\frac{T(n)}{n} = \frac{T(1)}{1} + \frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \cdots + \frac{1}{\lg n}$$

$$\rightarrow \frac{T(n)}{n} = T(1) + \Theta(\lg \lg n)$$

$$\rightarrow T(n) = \Theta(n \lg \lg n)$$

Answer

Note: the recurrence $T(n) = 4T(n/4) + n/lg\ n$ cannot be solved using master method.

Question 5

You are given k sorted arrays A_1, ..., A_k, with n elements each, and you want to suitably combine them into a single sorted array (with kn elements). Assume that each of the sorted arrays are put into another array (of arrays) B, with $B[i] = A_i$. One strategy for combining the arrays is as follows.

COMBINE(*B*[*i..j*]))
if (j-i = 0)
  return $B[i]$
$m = (i+j)/2$
X = *COMBINE*(*B*[*i..m*])
Y = *COMBINE*(*B*[*m+1..j*])
Merge X and Y and return the result

Let $T(k,n)$ be the time required to combine k arrays of length n each. Can you state the recursive formula for $T(k,n)$?

Question 5

Answer:

- Each problem with k arrays is divided into two subproblems with $k/2$ arrays each.
- The amount of time to merge two arrays is proportional to the total number of elements in the resulting array.
- Hence
  $T(k, n) = 2T(k/2, n) + Ckn$

Question 6

For the recursive formula you obtained in previous question, can you determine its asymptotically tight upper bound (in terms of both *k* and *n*)?

Answer

Answer: $T(k, n) = 2T(k/2, \mathrm{n}) + Ckn$

Recursion tree

- height = log k
- Work in each level = Ckn
- Total work = Ckn log k