2M Important problem types:

1) Sorting.
2) Searching
3) String processing
4) Graph problems
5) Combinatorial problems
6) Geometric problems
7) Numerical problems

- Numerical problems

16) What are the steps involved in the analysis framework?

The various steps are as follows

- Measuring the input's size
- Units for measuring running time
- Orders of growth
- Worst case, best case and average case efficiencies

17) What is the basic operation of an algorithm and how is it identified?

2

- The most important operation of the algorithm is called the basic operation of the algorithm, the operation that contributes the most to the total running time.
- It can be identified easily because it is usually the most time consuming operation in the algorithms innermost loop.

18) What is worst-case efficiency?

- The worst-case efficiency of an algorithm is its efficiency for the worst-case input of size n, which is an input or inputs of size n for which the algorithm runs the longest among all possible inputs of that size.
- Eg.: In linear search, $C_{worst}(n) = n$.

19) What is best-case efficiency?

of an algorithm is its efficiency for the best-case input of size

- One can decide the efficiency of particular algorithm on particular machine.
- The hypothesis for algorithm's efficiency class can be developed

2. ***Decide the efficiency metric M. Also decide the measurement unit.***

The efficiency of an algorithm can be measured by following different methods –

- Insert a counter in the algorithm in order to count the number of times the basic operation is executed. This is a straightforward method of counting an efficiency
2) example of algorithm.

For example : If we write a function for calculating sum of n number in an array then we can find the efficiency of that function by inserting a frequency count. The frequency count is a count that denotes how many times the particular statement is executed.

```
int sum element(int a[10],int n)
{
 int i, sum=0;
 for(i=0;i<n;i++)
 {
    sum=sum+a[i];
 }
 return sum;
}
```

The execution starts from for loop. The declaration part can be neglected. Now

Statement	Frequency count
i=0	1
i<n	This statement executes for (n+1) times. When condition is true i.e. when i<n is true, the execution happens to be n times and the statement executes once more when i<n is false.
i++	n times
sum=sum+ a[i]	n times
return sum	1
Total	(3n+3)

) ALGORITHM   Armstrongtest(n)
// problem description : This algorithm is find whether
// the given number is armstrong or not
// Input : The number to be tested that is n
// Output : appropriate message indicating

```
READ n
temp=n
sum=0
WHILE n>=0
    sum = sum + (n%10)*(n%10)*(n%10)
    n = n/10
ENDWHILE
IF sum == temp
    WRITE "NUMBER IS AN ARMSTRONG NUMBER"
ELSE
    WRITE "NUMBER IS NOT AN ARMSTRONG NUMBER"
```

Master Theorem:-

$T(n) = a^T(n/b) + \theta(n^k \log^p n)$

$a \geq 1, b > 1, K \geq 0$ and P is real number.

i) if $a > b^K$, Then $T(n) = \theta(n^{\log_b a})$.

(ii) if $a = b^K$

a). if $P > -1$, then $T(n) = \theta(n^{\log_b a} \log^{p+1} n)$

b). if $P = -1$, then $T(n) = \theta(n^{\log_b a} \log n)$

c) if $P < -1$, then $T(n) = \theta(n^{\log_b a})$

(iii) if $a < b^K$.

a) if $P \geq 0$, then $T(n) = \theta(n^k, \log^p n)$

b) if $P < 0$, then $T(n) = \theta(n^K)$.

General plan for Empirical Analysis of Algorithm:

1). understand the purpose of experiment of given algorithm.

2). Decide the efficiency metric M. Also decide the measurement unit. For ex,. operations's count Vs time

3). Decide on characteristics of the input.

4). create a program for implementing the algorithm this program is ready for experiment.

5). Generate a sample of input.

6). Run the algorithm for some set of input sample. Record the results obtained.

7). Analyze the resultant data.

* There are 3 major variation of decrease and conquer :

* decrease by a constant
* decrease by a constant factor
* variable size decrease

9

LINEAR SEARCH VERSUS BINARY SEARCH

LINEAR SEARCH	BINARY SEARCH
An algorithm to find an element in a list by sequentially checking the elements of the list until finding the matching element	An algorithm that finds the position of a target value within a sorted array
Also called sequential search	Also called half-interval search and logarithmic search
Time complexity is O(N)	Time complexity is O(log2N)
Best case is to find the element in the first position	Best case is to find the element in the middle position
It is not required to sort the array before searching the element	It is necessary to sort the array before searching the element
Less efficient	More efficient
Less complex	More complex

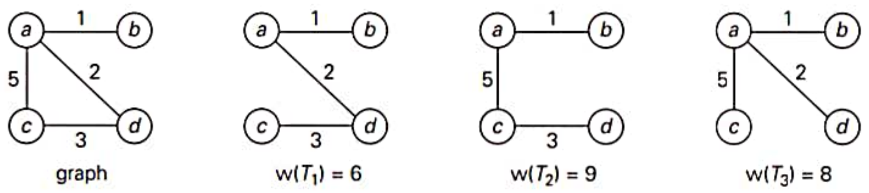


FIGURE 9.2 Graph and its spanning trees, with T_1 being the minimum spanning tree.

9.1 Prim's Algorithm

The following problem arises naturally in many practical situations: given n points, connect them in the cheapest possible way so that there will be a path between every pair of points. It has direct applications to the design of all kinds of networks—including communication, computer, transportation, and electrical—by providing the cheapest way to achieve connectivity. It identifies clusters of points in data sets. It has been used for classification purposes in archeology, biology, sociology, and other sciences. It is also helpful for constructing approximate solutions to more difficult problems such the traveling salesman problem (see Section 12.3). 12

We can represent the points given by vertices of a graph, possible connections by the graph's edges, and the connection costs by the edge weights. Then the question can be posed as the minimum spanning tree problem, defined formally as follows.

✓ **DEFINITION** A ***spanning tree*** of an undirected connected graph is its connected acyclic subgraph (i.e., a tree) that contains all the vertices of the graph. If such a graph has weights assigned to its edges, a ***minimum spanning tree*** is its spanning tree of the smallest weight, where the ***weight*** of a tree is defined as the sum of the weights on all its edges. The ***minimum spanning tree problem*** is the problem of finding a minimum spanning tree for a given weighted connected graph.

Figure 9.2 presents a simple example illustrating these notions.

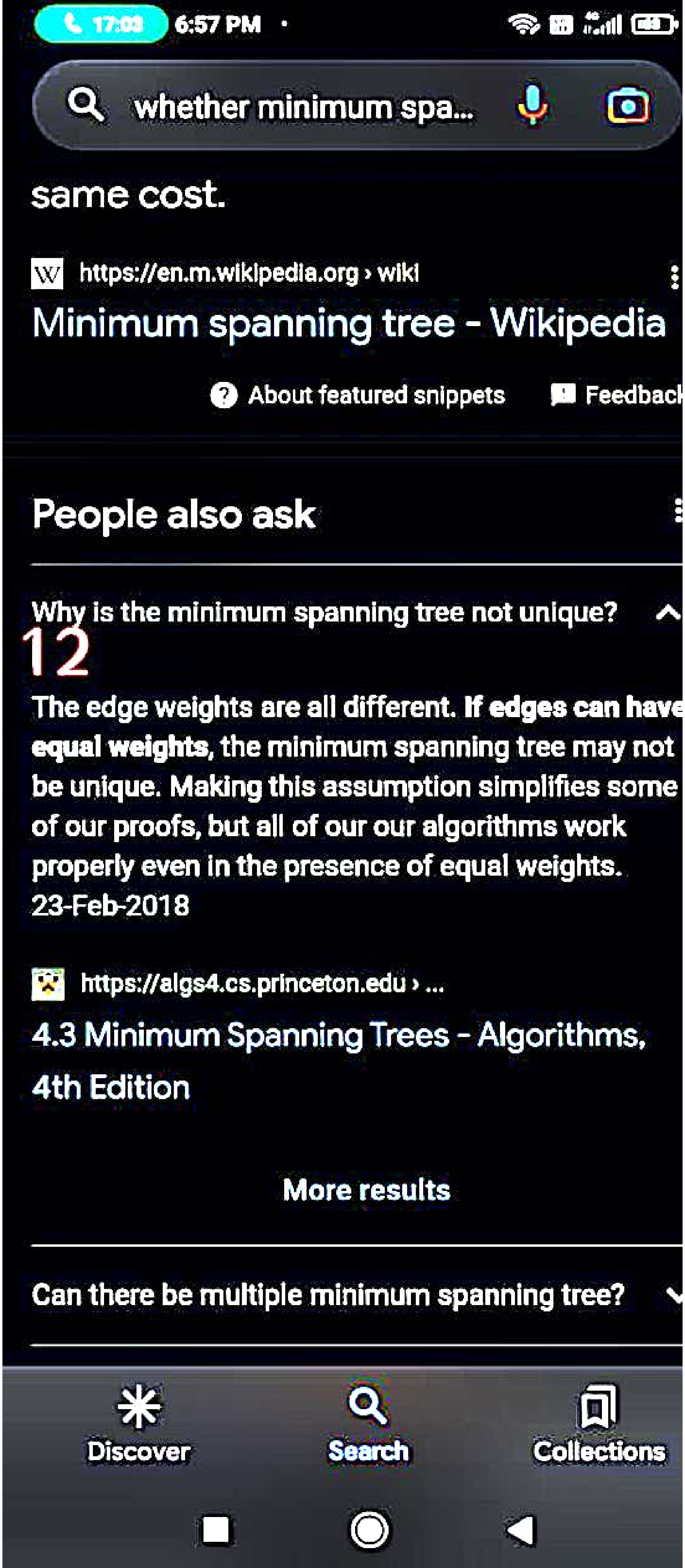
If we were to try constructing a minimum spanning tree by exhaustive search, we would face two serious obstacles. First, the number of spanning trees grows exponentially with the graph size (at least for dense graphs). Second, generating all spanning trees for a given graph is not easy; in fact, it is more difficult than finding a *minimum* spanning tree for a weighted graph by using one of several efficient algorithms available for this problem. In this section, we outline ***Prim's algorithm***, which goes back to at least 1957[1] [Pri57].

1. Robert Prim rediscovered the algorithm published 27 years earlier by the Czech mathematician Vojtěch Jarník in a Czech journal.

Prim's algorithm constructs a minimum spanning tree through a sequence of expanding subtrees. The initial subtree in such a sequence consists of a single vertex selected arbitrarily from the set V of the graph's vertices. On each iteration, the algorithm expands the current tree in the greedy manner by simply attaching to it the nearest vertex not in that tree. (By the nearest vertex, we mean a vertex not in the tree connected to a vertex in the tree by an edge of the smallest weight. Ties can be broken arbitrarily.) The algorithm stops after all the graph's vertices have been included in the tree being constructed. Since the algorithm expands a tree by exactly one vertex on each of its iterations, the total number of such iterations is $n - 1$, where n is the number of vertices in the graph. The tree generated by the algorithm is obtained as the set of edges used for the tree expansions.

Here is pseudocode of this algorithm.

ALGORITHM *Prim(G)*



17:03 6:57 PM

whether minimum spa...

same cost.

https://en.m.wikipedia.org › wiki

Minimum spanning tree - Wikipedia

About featured snippets Feedback

People also ask

Why is the minimum spanning tree not unique?

12

The edge weights are all different. **If edges can have equal weights**, the minimum spanning tree may not be unique. Making this assumption simplifies some of our proofs, but all of our our algorithms work properly even in the presence of equal weights.
23-Feb-2018

https://algs4.cs.princeton.edu › ...

4.3 Minimum Spanning Trees - Algorithms, 4th Edition

More results

Can there be multiple minimum spanning tree?

Asymptotic Notation and its properties:

To choose the best algorithm we need to check efficiency of each algorithm.

The efficiency can be measured by computing time complexity of each algorithm.

Asymptotic Notation is a short hand way to represent the time complexity.

Various notation such as

14

i) Big Oh Notation (O)

ii) Omega Notation (Ω)

iii) Theta Notation (Θ):

17

Applications of algorithm visualization

1. Research

- Many uncovered features of algorithm can be effectively shown by visual systems and researcher can get the benefit of such systems for further development and studies.

2. Education

- Students can learn algorithms very easily using animated systems. Various visual components included in the animated systems help the student to understand and analyze the algorithm in simplest manner. For example different sorting algorithms can be analyzed and understood with the help of bar charts or line graphs.