**Q.5. STRASSENS’s Matrix multiplication**

Practical implementation of Strassens’s matrix multiplication algorithm usually switch to the brute force method after matrix sizes become smaller than some crossover point. Run an experiment to determine such crossover point on your computer system.

**Ans-**

**Q.6. QUICK SELECT**

Use the **QUICK SELECT** algorithm to find **3rd largest element** in an array of n integers. Analyze the performance of **QUICK SELECT** algorithm for the different instance of size 50 to 500 element. Record your observation with the *number of comparison made* vs. *instance*.

**Q.7. Matrix Chain Multiplication Comparision**

Given a matrix chain A1 …An with the dimension of each of the matrices given by the vector **p** = <12,21,65,18,24,93,121,16,41,31,47,5,47,29,76,18,72,15>. (n=17) Write and run both the dynamic programming and memorized versions of this algorithm to determine the minimum number of multiplications that are needed (use type ***longint***) and the factorization that produces this best case number of multiplications. Run each of the two programs over an appropriately large number of times (put each in a loop to run repeatedly x times) and obtain the times at the beginning and end of the run. Use these times to determine the comparative runtimes of the two algorithms.

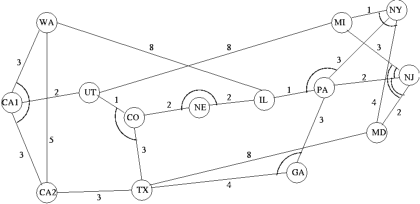
**Q.8. BINOMIAL COEFFICIENTS**

Computing the **binomial coefficients** *C*(*n*, *k*) defined by the following recursive formula:



Write the program with three different algorithm to compute **binomial coefficients** *C*(*n*, *k*) and compare them?

**Q.9. SPANNING TREE**

****

Write a program obtain minimum cost spanning tree the above NSF network using Prim’s algorithm, Kruskal’s algorithm and Boruvka’s algorithm..

**Q.10. 0-1 KNAPSACK PROBLEM**

Write a program that computes optimal solution to the 0–1 Knapsack Problem using dynamic programming? You may test your program with the following example:

There are *n* = 5 objects with integer weights *w*[1..5] = {1,2,5,6,7}, and values *v*[1..5] = {1,6,18,22,28}. Assuming a knapsack capacity of 11).

**Q.11. STRING MATCHING ALGORITHM**

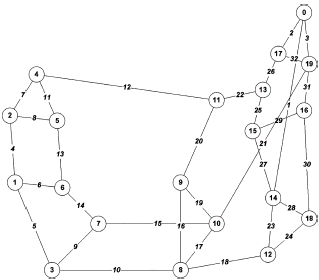
Given two strings P and T over the same alphabet set Σ, determine whether P occurs as a substring in T (or find in which position(s) P occurs as a substring in T). The strings P and T are called pattern and text respectively. Compare the efficiency of three string matching algorithms (Brute-Force Algorithm, Knuth-Morris-Pratt and Boyer-Moore Algorithm ) by varying pattern length [1-15] for n=5000.

**Q.12. MATRIX CHAIN MULTIPLICATION**

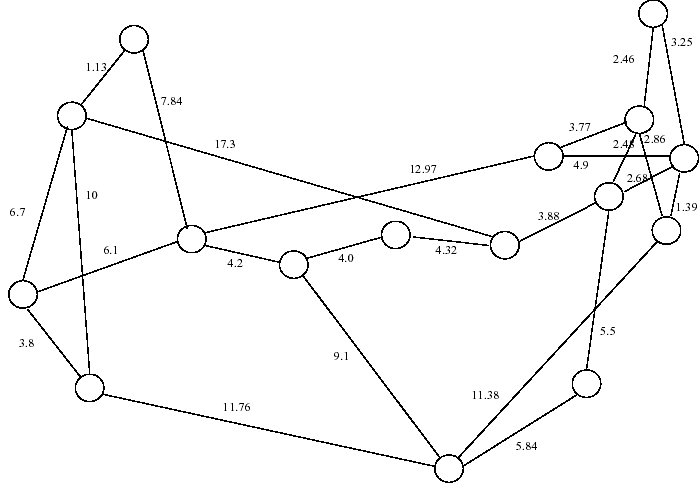
Write a program to compute the best ordering of matrix multiplication. Include the routine to print the actual ordering.

**Q.13. PATHING ALGORITHM**

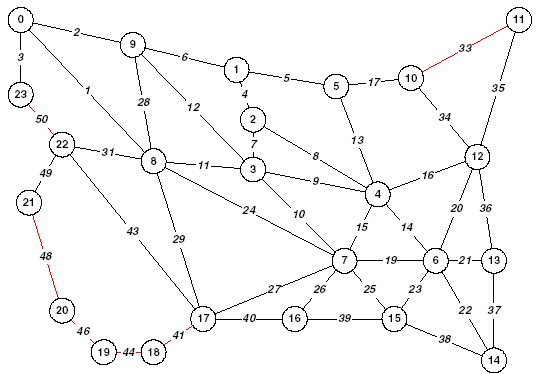
Use Floyd–Warshall algorithm (also known as Floyd's algorithm) to compute all pair shortest path for any one of the following standard network



ARPA Network



#### NSF Network



#### NATIONAL NETWORK