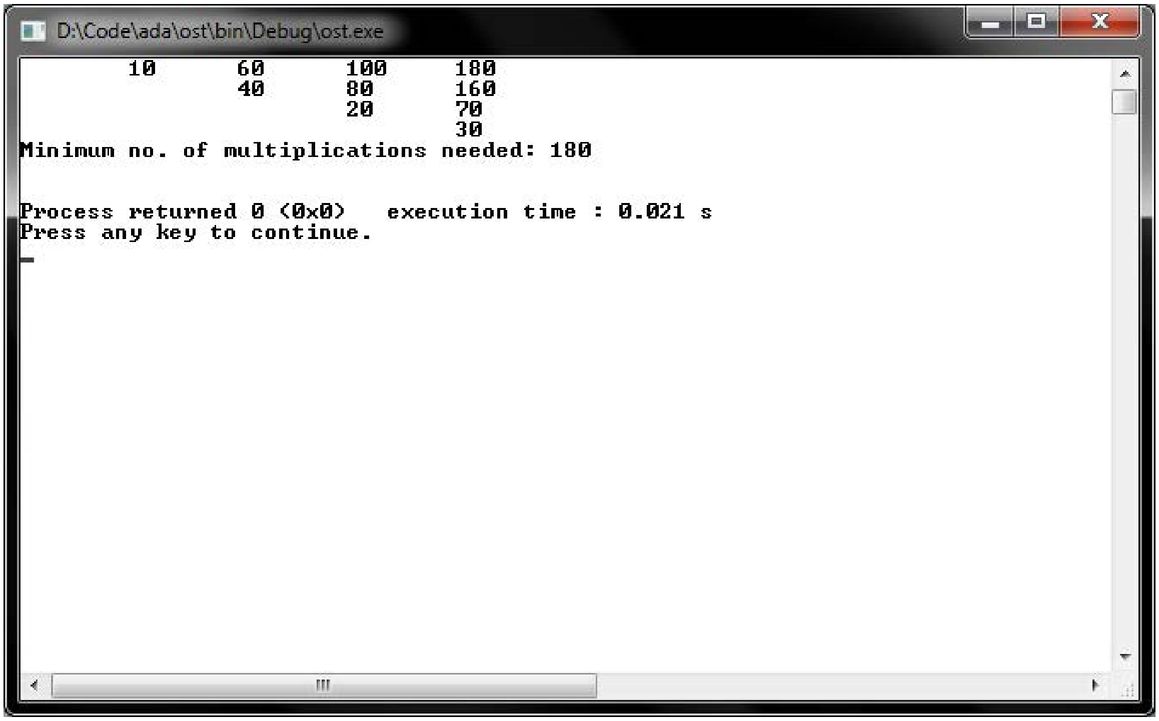
Optimal Binary Search Tree

```
#include <iostream>
#include <algorithm>
#include <vector>
#define LL long long int
using namespace std;
typedef vector<long long> vi;
typedef vector< vector<long long> > vii;
const LL inf = 1LL << 60;

void print(vii M) {…}

LL OST(vi& V) {
  vii M(V.size()+1,vi(V.size()+1,inf)), S(V.size()+1,vi(V.size()+1,0));
  LL n = V.size()+1;
  for(LL i=1; i<n; i++) {
    M[i][i] = V[i-1];
    S[i][i] = V[i-1];
  }
  for(LL i=1; i<n; i++) {
    for(LL j=i+1; j<n; j++) {
      S[i][j] = S[i][j-1] + V[j-1];
    }
  }

  for(LL l=1; l<=n; l++) {    //No of matrices to be multiplied, i.e range of j
    LL depth = n-l;        //Range till which i will go
    for(LL i=1; i<depth; i++) {
      LL j=i+l;
      for(LL k=i; k<=j; k++) {
        M[i][j] = min(M[i][j], (k<=i ? 0:M[i][k-1]) + (k>=j ? 0:M[k+1][j]) + S[i][j]);
      }
    }
  }
  print(M);
  return M[1][n-1];
}

int main() {
  vi V = {10,40,20,30};
  cout<<"Minimum no. of multiplications needed: "<<OST(V)<<"\n\n";
  return 0;
}
```

Output:



```
        10      60      100     180
                40      80      160
                        20      70
                                30
Minimum no. of multiplications needed: 180


Process returned 0 (0x0)   execution time : 0.021 s
Press any key to continue.
```