```cpp
/*** Here is a nice implementation of Dinitz blocking flow algorithm in C++.
(with special thanks to Fahim vai).
Works in undirected large graph containing multiple edges and self loops as well.
No STL used. This implementation is pretty fast.

Here, input is, number of nodes 2≤n≤5000, number of input edges 0≤e≤30000,
then e undirected edges in the form (u, v, cap) (1≤u,v≤n and 1≤cap≤10^9).
Source and Sink are assumed 1 and n accordingly, can be changed in the init() function call.

///N.B - Using adjacency matrix and/or STL makes it 10 to 4 times slower.
***/
#include<bits/stdc++.h>
using namespace std;

#define SET(p) memset(p, -1, sizeof(p))
#define CLR(p) memset(p, 0, sizeof(p))
#define i64 long long

const int INF = 0x7fffffff;
const int MAXN = 5005, MAXE = 60006;

int src, snk, nNode, nEdge;
int Q[MAXN], fin[MAXN], pro[MAXN], dist[MAXN];
int flow[MAXE], cap[MAXE], Next[MAXE], To[MAXE];

inline void init(int _src, int _snk, int _n) {
    src = _src, snk = _snk, nNode = _n, nEdge = 0;
    SET(fin);
}

inline void add(int u, int v, int c) {
    To[nEdge] = v, cap[nEdge] = c, flow[nEdge] = 0, Next[nEdge] = fin[u], fin[u] = nEdge++;
    To[nEdge] = u, cap[nEdge] = c, flow[nEdge] = 0, Next[nEdge] = fin[v], fin[v] = nEdge++;
}

bool bfs() {
    int st, en, i, u, v;
    SET(dist);
    dist[src] = st = en = 0;
    Q[en++] = src;
    while(st < en) {
        u = Q[st++];
        for(i=fin[u]; i>=0; i=Next[i]) {
            v = To[i];
            if(flow[i] < cap[i] && dist[v]==-1) {
                dist[v] = dist[u]+1;
                Q[en++] = v;
            }
        }
    }
    return dist[snk]!=-1;
}

int dfs(int u, int fl) {
    if(u==snk) return fl;
    for(int &e=pro[u], v, df; e>=0; e=Next[e]) {
        v = To[e];
        if(flow[e] < cap[e] && dist[v]==dist[u]+1) {
            df = dfs(v, min(cap[e]-flow[e], fl));
            if(df>0) {
                flow[e] += df;
                flow[e^1] -= df;
                return df;
            }
        }
    }
    return 0;
}
```

```
69
70  i64 dinitz() {
71      i64 ret = 0;
72      int df;
73      while(bfs()) {
74          for(int i=1; i<=nNode; i++) pro[i] = fin[i];
75          while(true) {
76              df = dfs(src, INF);
77              if(df) ret += (i64)df;
78              else break;
79          }
80      }
81      return ret;
82  }
83
84  int main() {
85      int n, e, u, v, c, i;
86      scanf("%d%d", &n, &e);
87      init(1, n, n);
88      for(i=0; i<e; i++) {
89          scanf("%d%d%d", &u, &v, &c);
90          if(u!=v) add(u, v, c);
91      }
92      printf("%lld\n", dinitz());
93      return 0;
94  }
```