#include <cstdio>

#include <algorithm>

using namespace std;

#define Max(a, b) ((a)>(b)?(a):(b))

int n, m, c;

int deg[1010];

struct edge

{

int p, q, w, lca;

} e[5010];

void read()

{

int i;

scanf("%d%d", &n, &m);

for (i = 0; i < m; i++)

{

scanf("%d%d%d", &e[i].p, &e[i].q, &e[i].w);

e[i].p--;

e[i].q--;

c += e[i].w;

}

}

int par[1010];

int color[1010], dep[1010], child[1010], finish[1010], l;

void calc\_tree(int p)

{

int q, i;

for (i = 0; i < m; i++)

if (e[i].w == 0)

{

q = -1;

if (p == e[i].p && color[e[i].q] == -1)

q = e[i].q;

if (p == e[i].q && color[e[i].p] == -1)

q = e[i].p;

if (q != -1)

{

color[q] = color[p]^1;

dep[q] = dep[p]+1;

child[q] = 1<<deg[p];

deg[p]++;

par[q] = p;

calc\_tree(q);

}

}

finish[p] = ++l;

}

void LCA()

{

int p, q, i;

for (i = 0; i < m; i++)

{

p = e[i].p;

q = e[i].q;

while (dep[p] < dep[q])

q = par[q];

while (dep[q] < dep[p])

p = par[p];

while (p != q)

{

p = par[p];

q = par[q];

}

e[i].lca = p;

}

}

bool operator < (edge a, edge b)

{

return finish[a.lca] < finish[b.lca];

}

int d[1010][1050];

int solve()

{

int s, l, p, q, u, v, i;

for (i = 0; i < m; i++)

if (e[i].w == 0 || color[e[i].p] == color[e[i].q])

{

s = e[i].w;

for (p = e[i].p, u=0; p != e[i].lca; u=child[p], p=par[p])

s += d[p][u];

for (q = e[i].q, v=0; q != e[i].lca; v=child[q], q=par[q])

s += d[q][v];

for (l = (1<<deg[e[i].lca])-1; l >= 0; l--)

if ((l & u) == 0 && (l & v) == 0)

d[e[i].lca][l] = Max(d[e[i].lca][l], s+d[e[i].lca][l | u | v]);

}

return d[0][0];

}

int main()

{

read();

memset(color, -1, sizeof(color));

color[0] = 0;

par[0] = -1;

child[0] = -1;

calc\_tree(0);

LCA();

sort(e, e+m);

printf("%d\n", c-solve());

return 0;

}