**Hopcropt Carp Maximum Bipertite Maching, O(E\*√V)**

**Match[], dist[] size = total left+right node**

**Tested on: spoj MATCHING**

bool bfs(int n)

{

int u, v, len;

queue< int > Q;

for(int i=1; i<=n; i++)

{

if(!match[i])

{

dist[i] = 0;

Q.push(i);

}

else dist[i] = 1e9;

}

dist[0] = 1e9;

while(!Q.empty())

{

u = Q.front();

Q.pop();

if(u)

{

len = adj[u].size();

for(int i=0; i<len; i++)

{

v = adj[u][i];

if(dist[match[v]]==1e9)

{

dist[match[v]] = dist[u] + 1;

Q.push(match[v]);

}

}

}

}

return(dist[0]!=1e9);

}

bool dfs(int u)

{

int v, len;

if(u)

{

len = adj[u].size();

for(int i=0; i<len; i++)

{

v = adj[u][i];

if(dist[match[v]]==dist[u]+1)

{

if(dfs(match[v]))

{

match[v] = u;

match[u] = v;

return true;

}

}

}

dist[u] = 1e9;

return false;

}

return true;

}

int hopcroft\_karp(int n)

{

int matching = 0;

memset(match,0,sizeof match);

while(bfs(n))

for(int i=1; i<=n; i++)if(match[i]==0 && dfs(i))matching++;

return matching;

}

int main()

{

int n, m, p, u, v;

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

for(int i=0;i<p;i++)

scanf("%d%d", &u, &v), adj[u].push\_back(n+v);

printf("%d\n", hopcroft\_karp(n+m));

return 0;

}