Undirected Graph

$G = \{V, E\}$ — V → vertices, E → edges; G → Graph

$V = \{v_1, v_2, v_3, v_4, \ldots, v_n\}$

$E = \{(v_i, v_j), (v_{i_2}, v_{j_1}), \ldots\}$

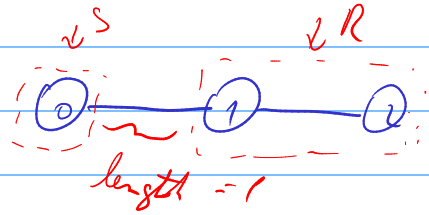
$E_{v_1, v_2} = E_{v_2, v_1}$ one there's no direction

Cut

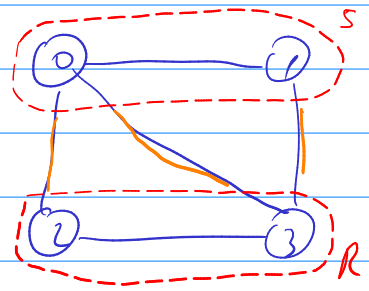
→ Partition of the vertex into 2 non-empty sets

S → non sub-set of vertices
$R = V - S$ → the remaining vertices

Size of a cut = the total of edges which connect a R_v with a S_v



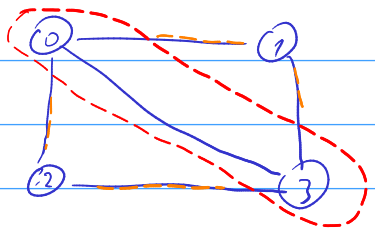
$S = \{0\}$ $R = \{1, 2\}$



$S = \{0, 1\}$ $R = \{2, 3\}$

Max-Cut

→ Maximize the size of a cut



$S = \{0, 3\}$ $R = \{1, 2\}$

size = 4

$f(0) = f(3) = 1$
$f(1) = f(2) = 0$

Decision Problem

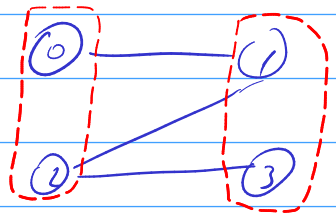
→ In this case, the idea is an algorithm that takes a graph and says whether there's a cut of size, at least, k.

→ We use Grover for that

→ For optimization, we can convert the problem in to a decision problem (binary search, for example) and run the algorithm in polynomial time (as in this case)

→ The max-cut is an optimization problem, but we can ask the algorithm if there's a cut of size 1, 2, 3, ... n and it says yes a lot as the answer. This way, we can convert the opt. to a single search and get a quadratic speedup from Grover's algorithm

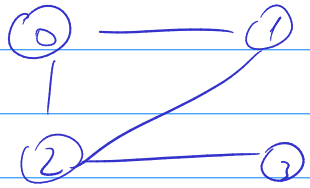
Bipartite graph



Max cut = $n(E)$

→ when node in S and R have no connection

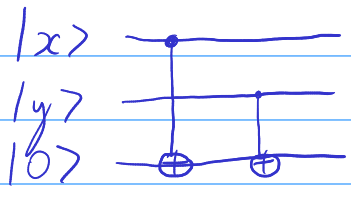
→ To have a speed up, we can ask if the graph is a bipartite graph



← not bipartite

Check edges

→ We can use a XOR to check if the edges came from the same set

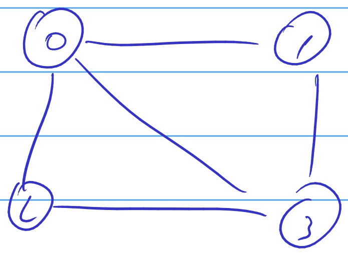


→ when 1 it's valid (different sets)

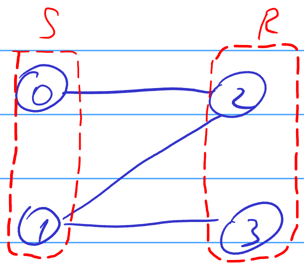
Oracle

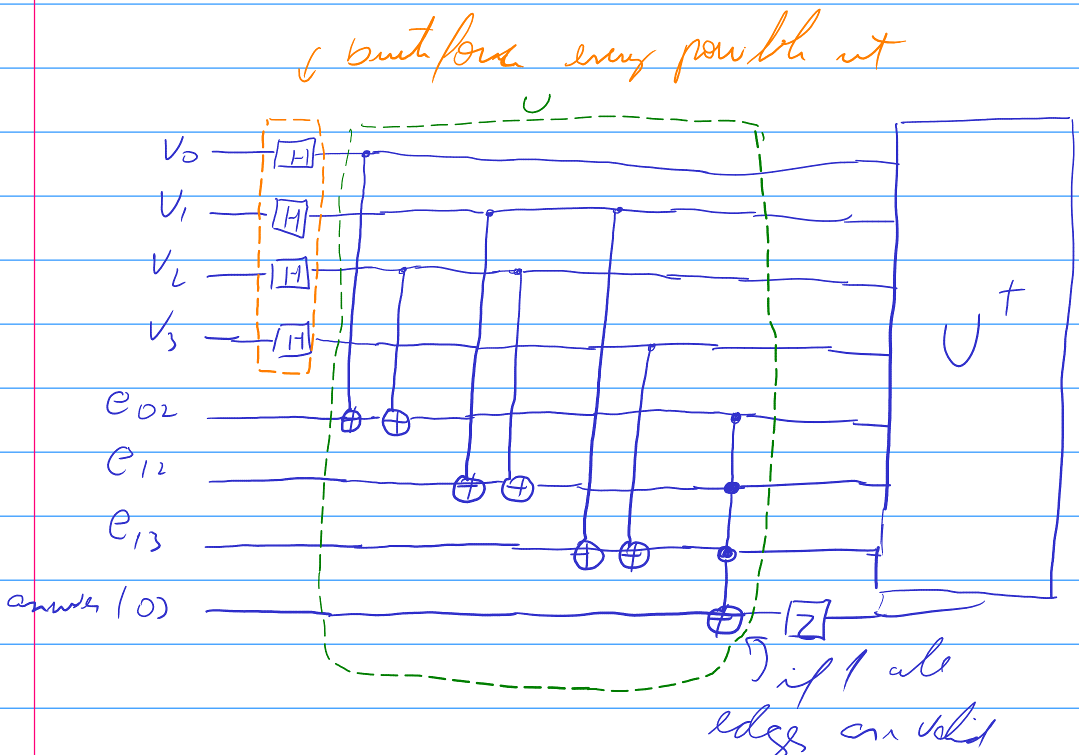
$|0\rangle \rightarrow V_i \in R$

$|1\rangle \rightarrow V_i \in S$



$|v_0 v_1 v_2 v_3\rangle \rightarrow |1001\rangle$





classical : $O(n^2)$ quantum : $O(\sqrt{2^n})$

in this example, the classical manner is already good enough

→ The greedy way of the book, after checking the edges, you need to run the results, then your answer quality will be based on the min k value you want.
→ You need different circuit for different K values