ADA HW4 b07902126 謝宗儒

PB.E

(1)

(1.1)

(w1->light, w4->dark, 8) + (w2->dark,w3->light,1) + (w1,w2,5) + (w3,w4,4) = 18

Both wizard increase 18 🡺totally increase 36

(1.2)  
(w4->light, w1->dark, 8) + (w4, w2->dark, 7) + (w4, w3->dark, 4) = 19

Both wizard increase 19 🡺totally increase 38

(2)

(2.1) Since each link will make one of the wands light and the other dark which equals to put the wands into two group. Therefore, it’s the same with finding a solution of a weighted maximum cut problem. We can regard the N magic wands as N vertex in a graph and the M fitness values as the corresponding edges with the weight = the fitness values to make the two problem equivalent.

To prove : maximum cut problem(MCP) ≤p weighted maximum cut problem(WMCP)

Reduction : given a MCP instance. If setting the weight of each edge = 1, obviously, the solution to cut the most edges on MCP will be equal to cut edges with the most sum of weight on WMCP.

Since MCP is NP-hard, this problem is also NP-hard by the derivation.

(2.2) Since the algorithm put each wand into either W1 or W2, and link each wand to all other wands in different set. It’s promised that the power enhanced in this solution (defined as S) >= (sum of M)/2.

Also, (sum of M) >= the power enhanced in optimal solution(defined as OPT) obviously.

Therefore, we get S >= (sum of M)/2 >= OPT/2.

🡺OPT/S <= 2 🡺this algorithm is a 2-approximation.

(3) Under RULE2, the number of dark wands should equal to light wands. That means the vertices at both sides of the cut should be the same.

To prove : weighted maximum cut problem(WMCP) ≤p weighted maximum equally cut problem(WMECP)

Reduction : given a WMCP instance. If adding N vertices and each vertex links to all other vertex with edge whose weight = 0. Since the adding vertices’ edge are with weighted = 0, there must exist many solutions of the WMCP. Then, since there are 2N vertices and N of them can be put to the both sides, we can absolutely find a solution from the solutions to let the number of vertices on the both sides are the same (as a solution of WMECP).

Since WMCP is NP-hard proved in (2.1), this problem is also NP-hard by the derivation.

(4)This problem ask to find minimum enhanced power with each wand exist at least one link. That means we should find minimum solution under the requirements the same with (3).

To prove : weighted maximum equally cut problem(WMECP) ≤p weighted minimum equally cut problem(WMinECP)

Reduction : given a WMECP instance and multiply each weight with -1 to be WMECP’. The solution of WMECP’ will obviously equal to WMinECP (the original instance). Therefore, WMinECP is equivalent to WMECP’.

Since WMECP(WMECP’) is NP-hard proved in (3), this problem is also NP-hard by the derivation.

(5)This problem can be regarded as a partition problem, and the Subset Sum Problem is equivalent to Knapsack problem.

Polynomial-time Reduction of Subset Sum Problem to this problem

= Polynomial-time Reduction of Knapsack Problem to partition problem

If the positive integer in Knapsack Problem is W and the checking result is yes.

Set H= sum(ti)/2, for i = 1 to N, and add t(N+1) = 2H + 2W, t(N+2)= 4H into the partition problem.

🡺sum(ti), for i=1 to N+2 = 2H + 2H+2W + 4H = 8H+2W

Since it’s promised that there exist a subset with sum\_value = W before adding t(N+1) and t(N+2), we can absolutely select out that subset and t(N+2) to make the new subset to be 4H+W. Then, the sum of remaining t will be

(2H-W) + (2H + 2W)=4H+W. Therefore, the partition problem is solved.

🡺proved Knapsack Problem ≤p Partition Problem.

PB.F (1)

(1-1)

Let Ci = the C right before the element of Si is inserted into C

If an element xk is inserted into C in the i iteration, the price = 1/|Si-Ci|, while |Si-Ci| is the number of covered x for this iteration.

When xk is about to be put in C, there are at least n-k+1 uncovered x,

and there must exist at least one S that can at least cover (n-k+1)/OPT

🡺|Si-Ci| >= (n-k+1)/OPT

🡺1/|Si-Ci| <= OPT/(n-k+1)

Proved.

(1-2)

Since the algorithm always pick up the S cover the must x as greedy choice, we can linearly count x in each S and sort them. The two action is promised finished in linear time. Therefore, the algorithm is in polynomial time.

Total price = number of iterations = |I| <= sum( OPT/(n-k+1 ), for k = 1 to n =Hn\*OPT

🡺ratio =|I|/OPT <= Hn = ln n

If n=1, ln n = 0 but actually time cost is O(1). Therefore, we should adjust the ratio with adding O(1) to avoid the error happened at n=1.

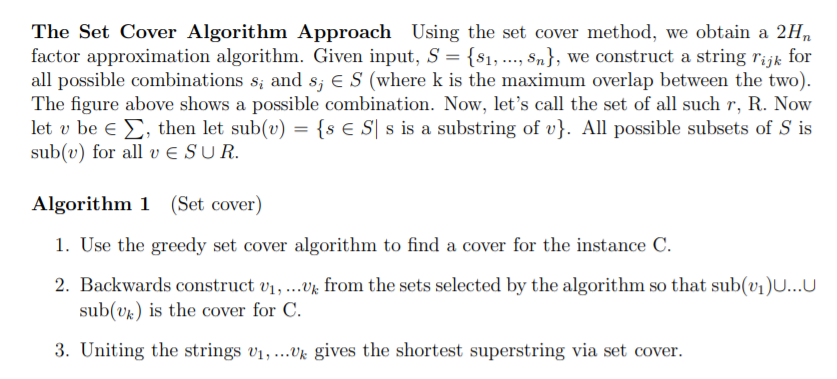
So, we will obtain a (ln n + O(1))-approximate solution.

(2)

(2-1)

Given an instance of this problem, we can regard each string in M as a node and a possible merged string as a cover-set covering some of M.

And follow the reference below.



(2-2)

Since the reduction in (2-1) is actually 2-approximate (Kasumi’s problem to Set Cover Problem). Also, it was proved in (1-2) that Set Cover Problem will obtain a

(ln n+ O(1) )-approximate solution. Therefore, Kasumi’s problem will obtain a

2\*(ln n + O(1) )-approximate solution = (2ln n + O(1))-approximate solution.