CS602 Applied Algorithms **Jul-Nov 2023**

Homework (No submission)

Matching

1. Describe an algorithm that finds a walk alternating between matching and non-matching edges, starting with a given source vertex and ending with a given destination vertex. A walk is a sequence of vertices where any two consecutive vertices are connected by an edge and repeating vertices is allowed.

2. Consider the problem of finding maximum weight matching in bipartite graphs. Suppose we have a matching of size i that has maximum weight among all matchings of size i. Suppose we find a minimum weight augmenting path (where weight of an augmenting path is defined as weight of matching edges minus the weight of non-matching edges). Prove that after swapping the edges, we will get a matching of size $i + 1$, which has maximum weight among all matchings of size $i + 1$.

   Use this idea to design an algorithm for maximum weight matching. How will you find a maximum weight augmenting path (in the bipartite case)? Is it a concern that some edges have negative weights? There are algorithms for shortest path when there are no negative weight cycles. Can we guarantee this here?

   If we try to apply this idea in the non-bipartite case with Edmonds Blossom idea, where will it fail?

3. Consider an undirected graph with edges colored either red or blue. We want to find a path from a given source to a given destination that alternates between red and blue edges. You can also specify the colors of starting and ending edges, that does not change the problem much. By definition, a path does not have repeated vertices. Design a polynomial time algorithm. Either you can try to modify Edmonds' algorithm or you can try to reduce this problem to finding maximum matching in a graph.

4. Consider the above red-blue path problem in directed graphs. Prove that it is NP-hard. Some related NP-hard problems, which may be useful, are – longest path, hamiltonian path, shortest path with negative and positive edge weights.

1. Describe an algorithm that finds a walk alternating between matching and non-matching edges, starting with a given source vertex and ending with a given destination vertex. A walk is a sequence of vertices where any two consecutive vertices are connected by an edge and repeating vertices is allowed.

Ans 1 we first build another directed graph G_1 where $e(a,b)$ iff $\exists\, c$ s.t. $e(a,c)$ is red and $e(c,b)$ is blue.
Now, on this graph, terminal states = red nhd of t or t.
and a simple DFS gives an alternating walk.
If not, construct G_2, $e(a,b)$ iff $\exists_c$ s.t. $e(a,c)$ is blue and $e(c,b)$ is red
$\|^{el}$ we get an alternating s-t walk.

Algorithm : Construct G_1 by iteration over all vertices
DFS(s, G_1)
Construct G_2 by " ——— "
DFS (s, G_2)

2. Consider the problem of finding maximum weight matching in bipartite graphs. Suppose we have a matching of size i that has maximum weight among all matchings of size i. Suppose we find a minimum weight augmenting path (where weight of an augmenting path is defined as weight of matching edges minus the weight of non-matching edges). Prove that after swapping the edges, we will get a matching of size $i+1$, which has maximum weight among all matchings of size $i+1$.

Use this idea to design an algorithm for maximum weight matching. How will you find a maximum weight augmenting path (in the bipartite case)? Is it a concern that some edges have negative weights? There are algorithms for shortest path when there are no negative weight cycles. Can we guarantee this here?

If we try to apply this idea in the non-bipartite case with Edmonds Blossom idea, where will it fail?

Ans 2 We have a maximum weight matching of size i, be M
minimum weight augmenting path. (matching - (non-match))
This gives a matching of size i+1,
Let maximum weight matching be M' (with size i+1)
Hence, $\exists$ augmenting path w.r.t M with edges of M'.
Now, to show this augmenting path has minimum weight.
Since M' is max-weight, (non-matching) − (matching) edges is max i.e. the weight of augmenting path is minimum.

3. Consider an undirected graph with edges colored either red or blue. We want to find a path from a given source to a given destination that alternates between red and blue edges. You can also specify the colors of starting and ending edges, that does not change the problem much. By definition, a path does not have repeated vertices. Design a polynomial time algorithm. Either you can try to modify Edmonds' algorithm or you can try to reduce this problem to finding maximum matching in a graph.

Ans 3

Method 1 : {incorrect}

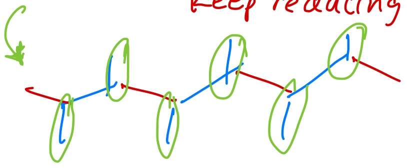
Find an alternating s-t walk using directed graph.
Then reduce that to a path by edmond's blossom method.

This method is incorrect as blossom formation relied on fact that entering edge is matching, neighs both non-matching, so no notion of "blossom".

Method 2 : (reduction to maximum matching)

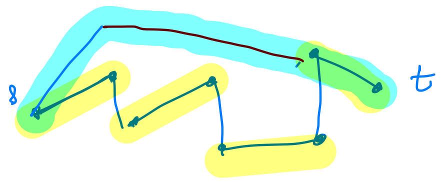
ques : how ~~to~~ do method 2 (take maximum red, blue matchings all s-v and t-u edges considered and take "intersection") ?

Idea : Take maximum matching of red, blue and keep "reducing" them using reverse augmenting paths



to reduce this issue

Idea : Take max blue matching, if s,t included, find red "crossings" if not induce augmentation using s/t/both → doesn't work because "important" region might not be covered here.

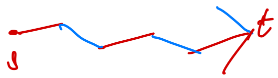


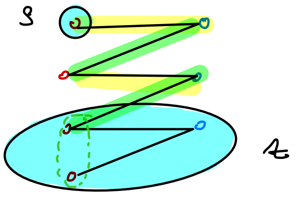
R-B alt path using max_match on some other graph ?

Can other graph be mono-colored ?/ some separation into r, b ?

(Brilliant alternative method)

Let E(G) be edge graph where only vertices joining red edge to blue edge are considered.

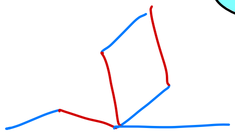




We want a path s.t.
in a given vertex set,
only two edges are allowed,
at s,t only one edge (node)
is allowed in the path

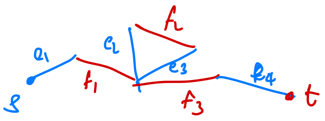
↓

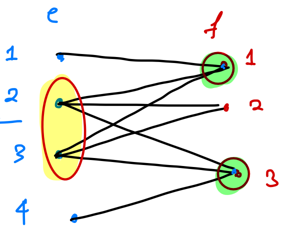
but vertex can
repeat, badly



(even length cycles
are ok)

#ask: how to get rid of
odd cycles in BFS.





such sets
for s,t
as well
at times.

two same color can
only occur once.

4. Consider the above red-blue path problem in directed graphs. Prove that it is NP-hard. Some related NP-hard problems, which may be useful, are – longest path, hamiltonian path, shortest path with negative and positive edge weights.

Ans4 Idea : Reduce SAT to this problem (atleast as hard as SAT). {or use other problems like hamiltonian path, TSP}

m1 :