Stewart Dulaney
SID: 904-064-791
CS 180

HW 8

7.22

a) An example of a matrix that is not rearrangable:

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Algorithm

b) This problem can be reduced to determining if a bipartite graph has a perfect matching.

We construct a flow network G' as follows:

- Let A be the vertices corresponding to rows and B be the vertices corresponding to columns
- Join $a_i \in A$ and $b_i \in B$ w/ a directed edge from LTR when the corresponding entry in the matrix is 1
- Add a source node s w/ a directed edge (s, a_i) for all $a_i \in A$
- Add a sink node t w/ a directed edge (b_i, t) for all $b_i \in B$
- Set the capacity of every edge to 1

Then run Ford-Fulkerson to find a maximum matching and then check if the matching is perfect (if the size of the maximum matching is equal to n).

Proof of Correctness

Claim: If we have a perfect matching, the matrix is rearrangable.

Pf: We know $\exists$ a map from n nodes to n nodes where no two nodes map to the same one. If we represent the graph as an adjacency matrix, for each row there is a 1 in each column s.t. no two rows have a 1 in the same column. Clearly, this can be rearranged to have 1's on the diagonal.

Claim: If the matrix is rearrangable, there is a perfect matching.

Pf: If the matrix can be arranged s.t. diagonal entries have 1's, there exists a mapping between rows and columns such that all n rows map to a unique column. This implies a matching w/ the same properties between the nodes in A and the nodes in B, and thus a perfect matching.

Time Complexity

Construction of a bipartite graph and transformation to a flow network takes poly. time. $O(n^2 + 2n^2) = O(n^2)$ The comparison to check for a perfect matching takes constant time $O(1)$. Ford-Fulkerson (FF) takes $O(mC)$ where $C = \sum_{e \text{ out of } s} c_e = |A| = n$ where $n = |A| = |B|$, so FF runs in $O(mn)$. So total is $O(n^2 + 1 + mn) = O(n^2 + mn)$.

7.23

Let $G = (V, E)$, $|V| = n$, $|E| = m$

Algorithm (EK)

Run the Edmonds-Karp Alg. on G to get max-flow f and the residual graph G_f. Run BFS from s on G_f to build the set U of nodes reachable from s. Next, reverse the graph (swap s and t, reverse all edges) to get G'. Run EK on G' to get max-flow f' and residual graph $G'_{f'}$. Run BFS from s on $G'_{f'}$ to build the set D of nodes reachable from s. Compute $C = V - U - D$ to get the central nodes. Return U, D, and C.

Proof of Correctness

Because the Edmonds-Karp Alg. uses BFS for path selection, it finds the shortest path w/ available capacity, and hence results in a min-cut (S, T) where S is of minimum cardinality. Hence, $U \subseteq A$ for all other min-cuts (A,B) and is the set of upstream nodes. Since we reversed the graph to obtain D, $D \subseteq B$ for all other min-cuts (A, B) and is the set of downstream nodes. It's easy to show the remaining vertices $C = V - U - D$ are central.

Time Complexity

Compute U: EK = $O(nm^2)$; BFS = $O(m+n)$

Compute D: EK = $O(nm^2)$; BFS = $O(m+n)$

Compute C: $O(1)$

Total = $O(2(nm^2 + m + n) + 1)$

= $\boxed{O(nm^2)}$

7.34 **Part (a)** We can reduce this to the max-flow problem.

Algorithm

There are 3 constraints:

1. each device must have k backup devices
2. each device can only be the backup device of b origin devices
3. each backup device must be within d meters of its origin device

We construct a flow network G as follows: Let $G = (V, E)$

- Add n nodes $d_{1,i}$ (one for each origin device) in a vertical line
- To the right of the nodes above, add n nodes $d_{2,j}$ (one for each backup device) in a vertical line
- For each node in the 1st column, compute the distance to each node in the 2nd column and connect them w/ a unit capacity edge from LTR (except do not connect a device with itself, when $i = j$) if that distance is $\leq d$
- Add a source node s and add a directed edge $(s, d_{1,i})$ of capacity k for each node $d_{1,i}$
- Add a sink node t and add a directed edge $(d_{2,j}, t)$ of capacity b for each node $d_{2,j}$

Next we run Ford-Fulkerson (FF) alg. to get the max-flow.

```
if max-flow = kn
    construct a map M of each device to its backup set
    return M
else
    return NULL
```

Proof of Correctness

Claim: If max-flow is kn, there is a valid backup set for all n devices.

- If max-flow is kn, then exactly k units of flow moved between the 1st column of nodes and the 2nd column, so for each of the n devices k backup devices with distance d (constraint 3) were found for each device.
- The edges w/ capacity k from s to $d_{1,i}$ ensure each backup set contains at most k devices (constraint 1)
- The edges w/ capacity b from $d_{2,j}$ to t ensure each backup device can only be associated w/ b origin devices (constraint 2)
- Thus if kn units of flow reached the sink, we got k units of flow from all n devices, so constraints for a valid backup set were satisfied for all n devices.

Claim: IF there is a valid backup set for all n devices, max-flow is kn.

- We will prove this by contrapositive.
- IF max-flow is $< kn$ (it cannot be $> kn$ b/c $\sum_{e \text{ out of } s} c_e = kn$),
  ∃ at least one of the n devices that produced $< k$ units of flow.
  Thus, the backup set found for that device has size $< k$, and
  ∴ there is not a valid backup set for all n devices.

Time Complexity

Construction of Flow network = $\underbrace{O(2 + 2n)}_{\#\text{ vertices}} + \underbrace{O(2n + n(n-1))}_{\#\text{ edges}} = \boxed{O(n^2)}$

FF Alg. $= O(|E| \cdot C)$ where $C = \sum_{e \text{ out of } s} c_e = kn$

$= O((2n + n^2 - n) \cdot (kn))$

$= \boxed{O(n^3)}$

Constructing maps of devices to backup sets = $\boxed{O(n^2)}$

Total $= O(n^2 + n^3 + n^2) = \boxed{O(n^3)}$

Algorithm

7.34 Part (b) We can still reduce to the max-flow problem w/ the new constraint.

First, we define some terms for simplicity:

- level-ℓ backup device: if device j is a backup for device i and the distance between i and j (δ) satisfies $p_\ell \le f(\delta) < p_{\ell-1}$ ($p_0 = +\infty$) then j is a level-ℓ backup of i; the highest level is level-1

Now the original problem can be rephrased as:

- Determine if each device can find at least one level-1 backup, at least two backups of of level-2 or above, ..., at least $k-1$ backups of level-$(k-1)$ or above, and at least k backups of level-k or above. No device can be in the backup sets of more than be other devices.

We construct a flow network G as follows: (see Figure 1) Let $G=(V,E)$

- For each device i, add $3k+1$ nodes: $v_{i\ell}$, $v_{i\ell}$, and $v'_{i\ell}$ for $\ell = 1,2,\ldots,k$ and v_i'
- Let S_ℓ be the subgraph containing all nodes $v_{i\ell}$ and $v'_{i\ell}$ for $i = 1,2,\ldots,n$
- Add source node s and sink node t
- For each node v_{i1}, add edge from s to v_{i1} w/ capacity k
- For each node $v_{i\ell}$, add edge from $v_{i\ell}$ to $v_{i\ell}$ w/ capacity $k+1-\ell$
- For each i, for $\ell = 1,\ldots,k-1$, add edge from $v_{i\ell}$ to $v_{i(\ell+1)}$ w/ capacity $k-\ell$
- For each pair (v_{i1}, v'_{j1}) s.t. $i \neq j$ in S_1, add a unit capacity edge from v_{i1} to v'_{j1} if the distance between them (δ) satisfies $f(\delta) \ge p_1$
- For each subgraph $S_2,\ldots,S_k$, for each pair $(v_{i\ell}, v'_{j\ell})$ s.t. $i \neq j$, add a unit capacity edge from $v_{i\ell}$ to $v'_{j\ell}$ if the distance between them (δ) satisfies $p_\ell \le f(\delta) < p_{\ell-1}$
- For each node $v'_{j\ell}$, add edge from $v'_{j\ell}$ to v'_j w/ capacity n
- For each node v'_j, add edge from v'_j to t w/ capacity b

Next we run Ford-Fulkerson (FF) alg. to get the max-flow.

```
if max-flow = kn
    construct a map M of each device to its backup set
    return M
else
    return NULL
```

Proof of Correctness

Claim: If max-flow is kn, there is a valid backup set for all devices.

- We let device j in device i's backup set if the edge from v_{ik} to v'_{jk} (in S_k) carries flow of 1 for some k.
- By the same reasoning as part (a), k backups are found for each device.
- Since $f^{out}_{v_{i(k-k+1)}} + f^{out}_{v_{i(k-k+2)}} + \dots + f^{out}_{v_{ik}} \leq k$ for each k, there are at least k-k backups of level k-k or above
- By the same reasoning as part (a), no device appears in more than b backup sets.

Claim: If there is a valid backup set for all devices, max-flow is kn.

- We let the edge from v_{ik} to v'_{jk} (in S_k) carry flow of 1 if device j is one of device i's level-k backups, so $f^{out}_{v_{i1}} + f^{out}_{v_{i2}} + \dots + f^{out}_{v_{ik}} = k$.
- To meet conservation and capacity constraints let each edge v_{ik} to v_{ik} carry flow $f^{out}_{v_{ik}}$, and each edge from v_{ik} to $v_{i(k+1)}$ carry flow $f^{out}_{v_{i(k+1)}}$, and each edge from S to v_{ik} carry flow k, and each edge v'_{jk} to v'_j carry flow of $\max(f^{in}_{v'_{jk}}) = n$, and each edge from v'_j to t carry flow of $\max(f^{in}_{v'_j}) = b$.
- Now we have a flow saturating all the edges out from s, which is a max-flow of kn.

Time Complexity

$$\text{Construction of } G = \underbrace{O(3kn + n + 2)}_{\#\text{ vertices}} + \underbrace{O(kn^2)}_{\max \#\text{ edges}} = \boxed{O(n^2)}$$

$$\text{FF Alg.} = O(|E| \cdot C) \text{ where } C = \sum_{e \text{ out of } s} c_e = kn$$
$$= O(kn^2 \cdot kn)$$
$$= \boxed{O(n^3)}$$

Constructing maps of devices to backup sets $= \boxed{O(n^2)}$

Note the backup set of device i can be built using the edges v_{ik} to v_{jk} in $S_1, \dots, S_k$ that carry flow of 1. There are at most $n(n-1)$ such edges in the whole graph.

$$\text{Total} = O(n^2 + n^3 + n^2) = \boxed{O(n^3)}$$

Figure 1. Construction for $k = 3$.