Midterm Solutions

Your name:

Open Book and Notes

(27) 1. Network Flow
Suppose we have C factories which produce cars and R factories which produce trucks. Let c_i represent the number cars per month the ith car factory can produce, and similarly r_i for the rth truck factory.

We also have D dealers each of whom sells both cars and trucks. dc_i, dr_i represent the number of cars, trucks the ith dealer wants to receive each month.

Each factory has a list of the dealers it can send its output to (so can send to some but not all).

a) Describe an efficient scheme to determine how many cars and trucks each factory should send to each dealer. If possible we want to supply each dealer with the desired number of cars and trucks. (draw a picture to explain your solution)

Start with a bipartite graph with C car factories and R truck factories on the left and $2D$ dealers on the right. Each dealer has two vertices; one for trucks and the other for cars. There exist an edge with a car factory and dealer's car vertex if that factory can supply cars to that dealer. The weight of this edge is the number of cars that the dealer wants.(It is still correct if the weight is larger, just not smaller.) Connect the truck factories to the dealer's truck vertices in the same manner as the cars. Create a source and a sink. Connect the source with all the factories, the weight of the edges is equal to that factories monthly output. Connect all the dealer vertices to the sink. If it is a dealer's car vertex then the weight of the edge is equal to the dealer's desired number of cars and likewise, if it a dealer's truck vertex, then the edge weight is the desired amount of trucks. Run a max-flow algorithm on the network. The amount of vehicles from a factor to a dealer is equal to the amount of flow from that factor vertex to the dealer vertex.

Many people only had one dealer vertex with incoming edges from the car and truck factories. The outgoing edge to the sink had a weight of $dc_i + dr_i$. This is wrong since the number of cars to the dealer may exceed dc_i.

Also, some people didn't give edge weights for all edges. ALWAYS give edge weights for all edges.

b) analyze the run time of your algorithm in terms of C,R and D.

b) Fifo-preflow push runs in $O(n^3)$ where

$n =$ vertices $= (C + R + 2D + 2) = O(C + R + D)$

$m =$ edges $= (C + R)D + C + 2D = O(CD + RD)$

Better bounds than this can be obtained (particularly by assuming the number of factories is much small than number of dealers), but was not expected.

(19) 2. Suppose you have an array of n elements: ABCD where A,B,C,D represent consecutive blocks of a, b, c, d elements respectively (e.g. if A is the first 20 elements, $a = 20$).

We want to convert the array to: ADCB using only constant extra space.

a) Describe a simple scheme in terms of algorithms we have studied to convert the array to ADCB. You may use routines described in the book or class without describing their details.

Reverse each of B,C and D. Then reverse the block $B^R C^R D^R$ (specific operations would be of the form: Reverse(a,a+b-1), ...

b) What is the running time of your solution? (you may use big O notation in terms of the sizes n, a, b, c, d given).

Each operation runs in linear time, so the total is $O(b + c + d)$

(13) 3. Suppose you have an n vertex m node flow network G where all arcs have capacity 1 or 2. Argue that if we run the generic pre-flow push algorithm we will get much better performance on this network than the general $O(mn^2)$ bound. State and justify your improved time bound for this setting.

An arc of capacity 1 only does a saturating push. For an arc (i, j) of capacity 2 there can be a non-saturating push (of value 1) but the next push is saturating. Before we can do another non-saturating push from i to j we must do a push from j to i. Thus between each two non-saturating pushes from i to j the height of i must increase by at least 2 (just as for saturating pushes). Thus we get the same $O(mn)$ bound for non-saturating pushes we got for saturating pushes, and the entire algorithm is then $O(mn)$.

(31) 4. Hashing:
a When implementing double hashing, some experiments have shown improved performance by using a second hash function which uses a mask and shift to extract two bits from the middle of the key and then adds one to their total (so the second hash function always has value 1 to 4).

Explain why this might improve performance for moderate load factors as compared to the second hash function ($h(x) = (\ x\ mod(m-1)\ )\ +1$). (hint: there are at least two good reasons).

1) Since values are 1-4 we get good locality for our probes.

2) Mask and shift are faster than mod, so compute it faster.

Note that this second hash function WILL likely cause more collisions, but each one is dealt with faster.

b Suppose we wanted to add deletions to our chaining program. Describe (at a high level) how to delete a key x from our table (you should also be able to handle attempting to delete a key not in the table).

Hash to find the right chain. Now look for x.

If found in table: if next pointer is nil, change value to -1, else, get value from first element in the chain, put in table, update pointer, and free the old structure.

If x not in the table, and pointer non-nil: do a LL search for x keeping a trailing pointer. When found, remove the structure in the LL containing x and free the structure.

c Estimate the time to do a delete compared to the time for an insert.

Key facts: a delete will usually be successful at finding x, and insert will usually not find a duplicate x value (but must look for it). Thus an insert is like an unsuccessful search (looks at whole chain), while a delete is like a successful search (looks at half the chain).

Also, delete has to keep two pointers (so a slightly slower search), and does a free rather than a malloc (probably faster).

(10) 5. Estimating:

A traveling salesperson claims to have driven 500,000 miles last year. Do you believe this?

To drive 500000 miles in a year is equal to 57 miles per hour. But that is 24hours/7 days a week. Since our salesman needs to sleep, eat, get gas and sell, he is unlikley to drive an average of even 16 hours per day. That would give an average speed of 85 MPH (some of it city driving). This is quite unbelievable.