CS 241 Algorithms and Data Structures **Spring Semester, 2003**

Solutions to Practice Problems for Homework 5 and Final

In the problems in which you are asked to give a bfs, dfs, or shortest path tree, instead of drawing it graphically, I will just list the parent pointer for each vertex. However, if I were hand writing these solutions then I would draw it as a tree. You should be able to easily move between these two representations.

1. (a) The order in which the vertices are visited by BFS are: A,C,E,B,F,D. In the BFS tree, A is the root, C,E, and B have A as their parent, F has E as its parent, and D has B as its parent.

   (b) The order in which the vertices are visited by DFS are: A,C,E,F,B,D. Here are the discovery and finishing times:

vertex	A	B	C	D	E	F
discovery time	1	8	2	9	3	4
finishing time	12	11	7	10	6	5

   (c) A valid topological order is obtained from placing the vertices in the reverse order of finishing time. Hence we get the order: A,B,D,C,E,F

2.

vertex	S	A	B	C	D	E	F	G
distance	0	5	13	4	12	10	13	7
parent	null	C	G	S	G	S	S	A

3. When A is used as the source vertex, the edges are added to the MST by Prim's algorithm in the following order: (A,C), (A,B), (B,D), (D,E), (E,F), (F,H), (G,H)

4. (a) Consider the row in the adjacency matrix corresponding to vertex u (denote it by $r_1,\ldots,r_n$ and the column of the adjacency matrix corresponding to vertex v (denote it by $c_1,\ldots,c_n$). For each i $(1 \le i \le k)$ if $r_i \wedge c_i$ is 1 then $(u,k) \in E$ and $(k,v) \in E$. Thus in $O(n)$ time we can solve the given problem.

   (b) With an adjacency list representation, we must consider all vertices adjacent to u. Then for each such vertex (k), we must go through k's adjacent list to see if v is there. In the worst case, all m edges of the graph may need to be examined. Hence the time complexity is $O(m)$. (Note that the time complexity here is independent of n).

5. (a) Just use Dijkstra's shortest path algorithm with the firestation as the source. Here's the outcome.

vertex	dist	parent	shortest path from G
G	0	null	G
A	12	C	G $\rightarrow$ E $\rightarrow$ D $\rightarrow$ C $\rightarrow$ A
B	6	H	G $\rightarrow$ H $\rightarrow$ B
C	8	D	G $\rightarrow$ E $\rightarrow$ D $\rightarrow$ C
D	5	E	G $\rightarrow$ E $\rightarrow$ D
E	2	G	G $\rightarrow$ E
F	8	G	G $\rightarrow$ F
H	3	G	G $\rightarrow$ H

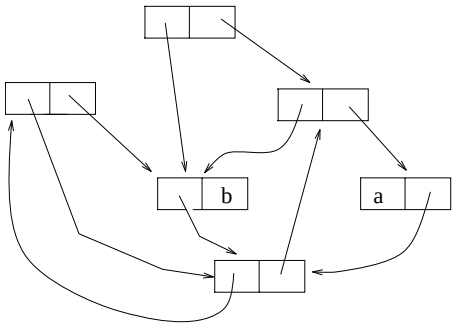
(b) Run Dijkstra's algorithm with each vertex as the source. For each remember the maximum key extracted from extractMin and then select the one with the smallest maximum key. Since you just need to run Dijkstra's algorithm f times, the time complexity is $f \cdot O(r + f \log f)$ when using Fibonacci heaps. THus the overall time complexity is $O(rf + f^2 \log f)$.

6. (a) Use a binary heap (or Fibonacci heap) with the time stamp as the key. It supports both required operations in $O(\log n)$ time with a small constant hidden in the big-oh notation.

   (b) Use a skiplist (with a handle returned at insert). This will allow an $O(1)$ time method to give the event that precedes it and the event that follows it and also to do the remove because no search is needed for the item since the handle is used.

   (c) Use a B-tree since it will minimize the number of disk pages read.

7. Here is memory as it began.



8. (a) Cell 0 was $[b, 2]$, cell 2 was $[a, 0]$, cell 3 was $[5, 0]$, and cell 5 was $[0, 2]$.

   (b) After Phase 1 we have the following where cell 13 is the start of the free list.

0	11	2
1	6	d
2	12	0
3	9	0
4	a	7
5	10	2
6	7	8
7	x	y
8	0	4

9	5	0
10	0	2
11	b	2
12	a	0
12	16	11
14	16	9
15	x	y
16	a	12
17	k	1

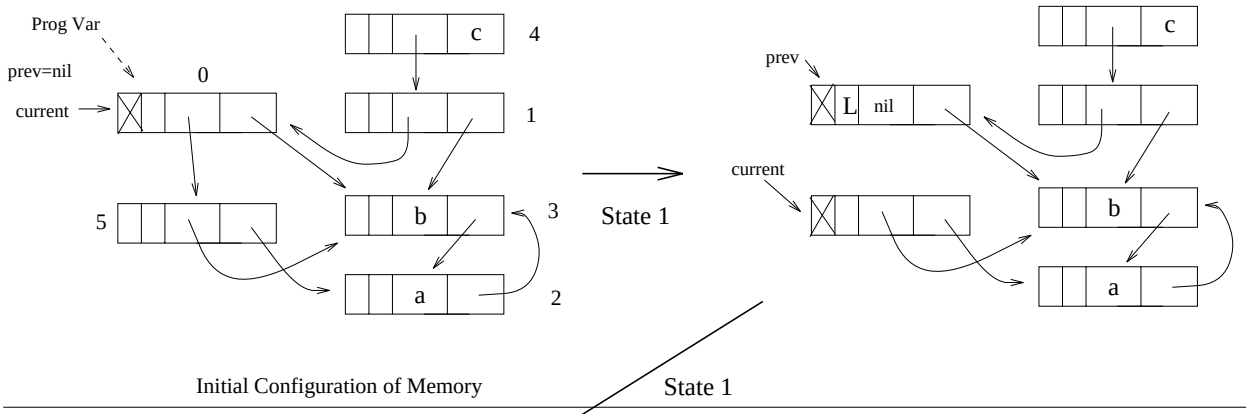
   (c) After Phase 2 we have the following where cells 0-8 are now the to-space and cells 9-17 are the from-space.

0	11	2
1	6	d
2	12	0
3	9	0
4	a	7
5	10	2
6	7	8
7	x	y
8	0	4

9	10	11
10	11	12
11	b	12
12	a	11
12	16	11
14	16	9
15	x	y
16	a	12
17	k	1

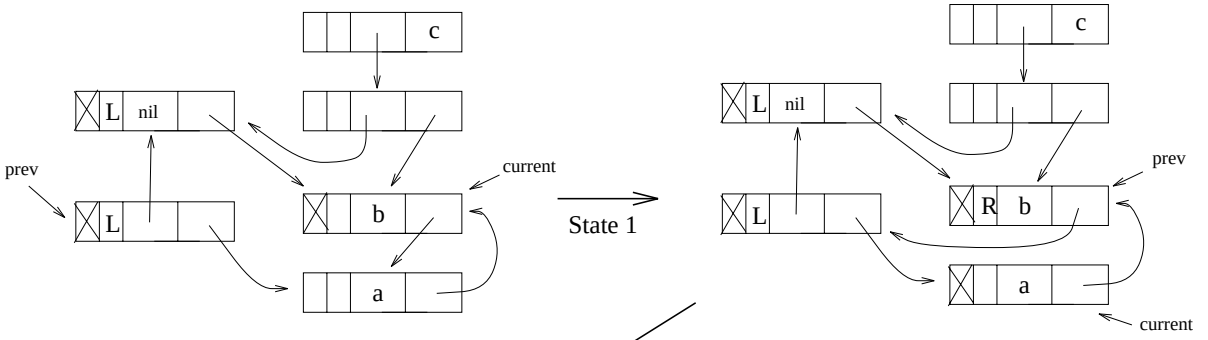
   (d) The cells currently available are 13-17.

9. Here's a simulation through the in-place DFS.

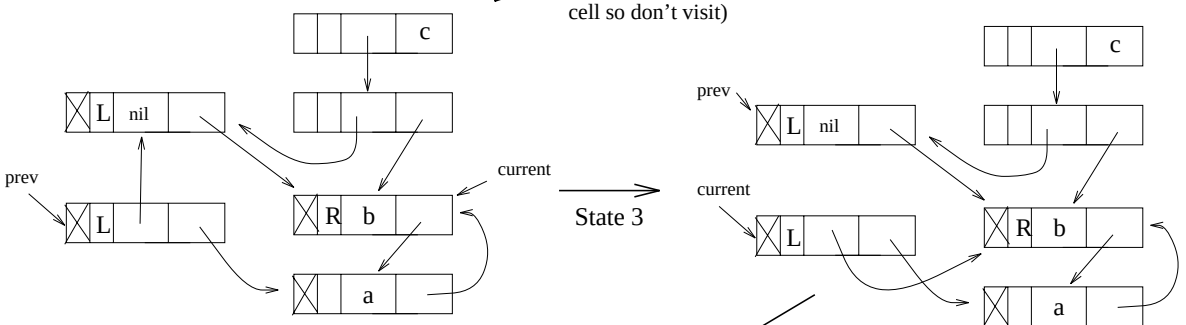


Initial Configuration of Memory

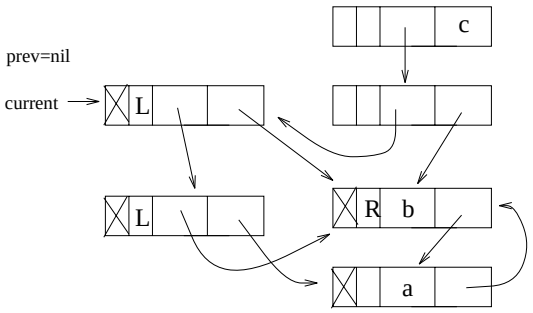
State 1



State 3
(pointer goes to marked
cell so don't visit)



State 3



DONE: Notice that cells 1 and 4 are garbage.