NPTEL MOOC,JAN-FEB 2015
Week 5, Module 3

DESIGN AND ANALYSIS OF ALGORITHMS

Priority queues

MADHAVAN MUKUND, CHENNAI MATHEMATICAL INSTITUTE
http://www.cmi.ac.in/~madhavan

Job scheduler

- A job scheduler maintains a list of pending jobs with their priorities.

- When the processor is free, the scheduler picks out the job with maximum priority in the list and schedules it.

- New jobs may join the list at any time.

- How should the scheduler maintain the list of pending jobs and their priorities?

Priority queue

- Need to maintain a list of jobs with priorities to optimise the following operations
  - delete_max( )
    - Identify and remove job with highest priority
    - Need not be unique
  - insert( )
    - Add a new job to the list

Linear structures

- Unsorted list
  - insert( ) takes O(1) time
  - delete_max( ) takes O(n) time
- Sorted list
  - delete_max( ) takes O(1) time
  - insert( ) takes O(n) time
- Processing a sequence of n jobs requires O(n^2) time

Two dimensional structures

N = 25

First attempt

- Assume N processes enter/leave the scheduler
- Keep an √N x √N array
- Each row is maintained in sorted order

12	17	29	31	40
8	19	22	33	37
10	13	14		
13	20	25	43	
6	11			

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	13	14			3
13	20	25	43		4
6	11				2

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time O(√N)

11

12	17	29	31	40	5
8	19	22	33	37	5
10	13	14			3
13	20	25	43		4
6	11				2

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time O(√N)

11

12	17	29	31	40	5
8	19	22	33	37	5
10	13	14			3
13	20	25	43		4
6	11				2

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time $O(\sqrt{N})$

	12	17	29	31	40	5
	8	19	22	33	37	5
11	10	13	14			3
	13	20	25	43		4
	6	11				2

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	**11**	13	14		3
13	20	25	43		4
6	11				2

insert( )

Insert 11

- Insert into first row that has free space
  - Maintain size of each row
- Takes time O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	**11**	13	14		**4**
13	20	25	43		4
6	11				2

delete_max( )

- Maximum in each row is the last element
- Maximum among these is to be deleted
- Again O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	11	13	14		4
13	20	25	43		4
6	11				2

delete_max( )

- Maximum in each row is the last element
- Maximum among these is to be deleted
- Again O(√N)

12	17	29	31	**40**	5
8	19	22	33	**37**	5
10	11	13	**14**		4
13	20	25	**43**		4
6	**11**				2

delete_max( )

- Maximum in each row is the last element
- Maximum among these is to be deleted
- Again O(√N)

12	17	29	31	**40**	5
8	19	22	33	**37**	5
10	11	13	**14**		4
13	20	25	**43**		4
6	**11**				2

delete_max( )

- Maximum in each row is the last element
- Maximum among these is to be deleted
- Again O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	11	13	14		4
13	20	25			4
6	11				2

delete_max( )

- Maximum in each row is the last element
- Maximum among these is to be deleted
- Again O(√N)

12	17	29	31	40	5
8	19	22	33	37	5
10	11	13	14		4
13	20	25			3
6	11				2

Two dimensional structures

Summary

- insert( ) takes $O(\sqrt{N})$
- delete_max( ) takes $O(\sqrt{N})$
- Processing N jobs takes $O(N\sqrt{N})$

Can we do better?

12	17	29	31	40
8	19	22	33	37
10	13	14		
13	20	25	43	
6	11			

Trees

- Maintain a special kind of binary tree called a heap
    - Balanced: N node tree has height log N
- Both insert( ) and delete_max( ) take O(log N)
    - Processing N jobs takes time O(N log N)
- Truly flexible, need not fix upper bound for N in advance