

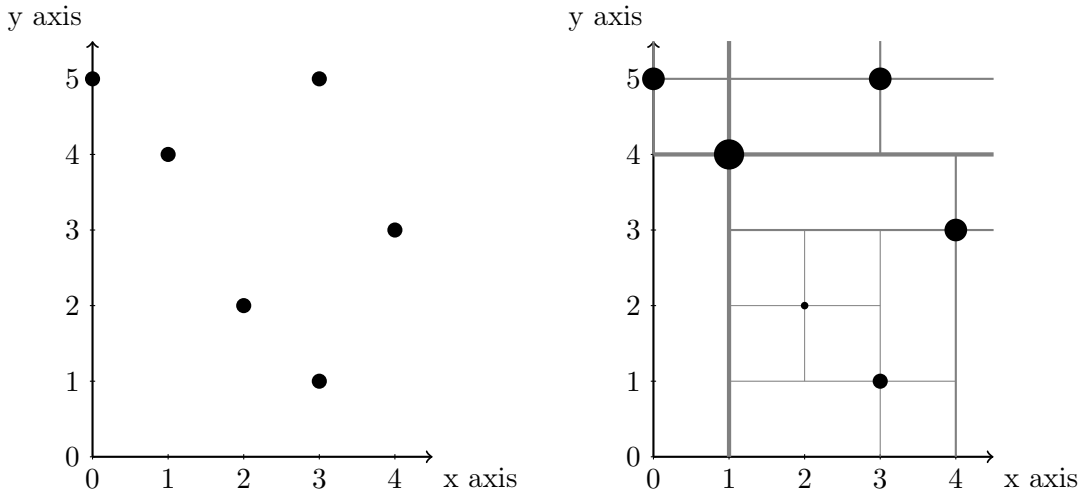
1 Quadsires

REMINDER FROM ASSIGNMENT 4 (up to the end of the page): The popular augmented reality game Lokimon Go maintains a large map of sites of interest—like Lokistops and Gyms—each with x and y coordinates. (Longitude and latitude, but for our purposes, we'll take these to be integer x and y on a Cartesian grid.)

These are stored in a type of "quad tree" data structure. Specifically, the structure is a 4-ary tree. Each node $=n=$ is associated with a Lokimon Go site of interest $=n.site=$, represented simply as a (x, y) coordinate pair giving the location of the site ($=n.site=$ for the pair or $=n.site.x=$ and $=n.site.y=$ for the individual coordinates). The node's four subtrees are $=n.NE=$, $=n.SE=$, $=n.SW=$, and $=n.NW=$. These are (respectively): the area to the northeast (higher x and y values), southeast (higher x and lower y values), southwest (lower x and y values), and northwest (lower x and higher y values). Some of these subtrees may be empty (represented by the special quad tree value $=EMPTY=$) if they contain no sites of interest.

There are some boundary cases: No two sites have the **same** coordinates. Sites that are due north or due east are both included in the $=NE=$ subtree, due south in the $=SE=$ subtree, and due west in the $=NW=$ subtree.

For example, on the left below is a set of sites we might want to put into a Lokimon Go quad tree. On the right is that set in a possible quad tree. The root is the largest dot at $(1, 4)$ with lines separating out its NE, SE, SW, and NW quadrants. It has three non-empty subtrees, and each root of these subtrees is drawn with a slightly smaller dot. The NE is rooted at $(3, 5)$, the SE at $(4, 3)$, and the NW at $(0, 5)$. Only the SE subtree has further children. It has a SW subtree (rooted at $(3, 1)$) which in turn has a NW subtree (rooted at $(2, 2)$).



We could then search for a site like $(2, 2)$ by accessing the root, which is at $(1, 4)$, observing that $(2, 2)$ is to the SE of the root, and recursing into the SE subtree of the root.

FURTHER REMINDER: We add to each node a non-negative integer $=freq=$ field representing the frequency at which we expect that node's site to be accessed in Lokimon Go. We want higher-frequency nodes to be closer to the root of the tree. In particular, if a node is k steps from the root (where for the root, $k = 0$), then we assign it a cost of $k \cdot freq$. The cost of a whole tree is the sum of the costs of each of its nodes.

...

Given a set of sites and their frequencies, we can ask which is the optimal quadtree (i.e., a quadtree that has minimal cost among all possible quadtrees built from the same sites).

2 Very Busy Printers

REMINDER FROM ASSIGNMENT 3:

The CS department buys a single 3-D printer and wants to develop a scheduling algorithm for it. We'll call their problem the Very Busy 3-D Printer Problem or VB3.

The input for a particular week is a list of n jobs for the printer. Each job is a pair of a positive integer duration t of the job and non-negative integer deadline d for the job. Once started, a job must be run to completion, which takes t minutes. To be useful, the job must be complete by the deadline d (also in minutes from the start of the week).

The output is a schedule: a list of $k \leq n$ of the jobs along with a non-negative integer start time s for each one, sorted in increasing order of start time. A valid schedule must ensure that no two jobs overlap (i.e., no jobs i and j exist in the schedule such that $s_i \leq s_j$ and $s_i + t_i > s_j$) and all jobs finish on time ($s_i + t_i \leq d_i$). Note that it is OK for a job's end time to match the next job's start time. The goal is to schedule as many of the jobs as possible.

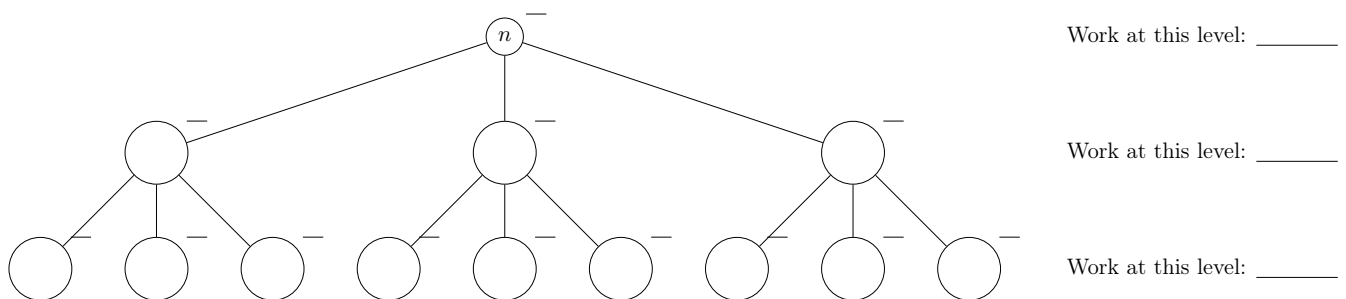
For example, the input $(2, 5), (1, 3), (3, 4), (5, 15)$ represents four jobs. The first is of length 2 minutes with a deadline 5 minutes from the "zero" time. The second job is only 1 minute long and has a deadline 3 minutes after the zero time. Etc.

The optimal solution to this instance includes three of the four jobs. One option—represented with tuples of (s, t, d) start time, duration, and deadline—would be $(0, 2, 5), (2, 1, 3), (8, 5, 15)$. That runs the 2 minute job right away, followed by the one minute job. Then, at minute 8, it runs the 5 minute job. This would also be an optimal schedule with different jobs and timing: $(0, 1, 3), (1, 3, 4), (4, 5, 15)$.

3 Recurrence Trees

In this problem, you complete recurrence trees for two recurrence relations, writing problem size and work in terms of n on a template. Specifically, we give you a sketch of the first two levels of a recurrence tree with **more nodes than needed** for the recurrence. You should: 1. complete only the leftmost nodes needed at each level, 2. write **problem size inside the node**, 3. write the **amount of work at that node in the blank** above-right of the node, 4. write the **work per level in the blank on the far right** of each level, 5. put an **X through nodes in the template that are not needed**, and finally 6. write a **good big-O bound on the height of the tree** below the tree. Since the root node has problem size n regardless of the form of the recurrence, we have filled that node in for you. Now, complete the tree for this recurrence:

$$T(n) = \begin{cases} 1 & \text{when } n \leq 0 \\ T(\sqrt{n}) + T(\lg n) + n^2 & \text{otherwise} \end{cases}$$



Height of the tree $\in O(\underline{\hspace{1cm}})$