CS41 Lab 9

November 2, 2021

The learning goals of lab this week are (i) to continue to practice building DP algorithm design skills, and (ii) to start thinking about flow problems and applications of network flow. I encourage you to work on the first problem and then whichever problem looks interesting.

1. **Pretty-printing** (based on KT 6.6). Suppose we have a paragraph of text, and we want to print it neatly on a page. The paragraph consists of a list of words $w_1, w_2, \ldots, w_n$; each word w_i has length ℓ_i. The maximum line length is M. (Assume that $\ell_i \leq M$ for all i.) We assume we have a fixed-width font and ignore issues of punctuation and hyphenation.

   Consider a line containing words $w_i, w_{i+1}, \ldots, w_j$, and using only one space between words. Because the words must fit within the maximum line length, we know that:

   $$\text{length of this line } = (\ell_i + 1) + (\ell_{i+1} + 1) + \cdots + (\ell_{j-1} + 1) + \ell_j \leq M$$

   The "slack" space on a line is the number of spaces remaining at the right margin, so for this line it is the value:

   $$\text{slack of this line} = M - \Big((\ell_i + 1) + (\ell_{i+1} + 1) + \cdots + (\ell_{j-1} + 1) + \ell_j\Big)$$

   The penalty is the sum over all the lines (including the last) of the *squares* of the slack of all lines in the paragraph.

   Describe and analyze a dynamic programming algorithm to find the best way to print a paragraph, where "best" means "with smallest penalty". Include a recursive definition of the optimal value that motivates your algorithm.

   For example, consider the following text.[1]

   With maximum line length 75, the output should look like:

   ```
   Not far from here, by a white sun, behind a green star, lived the
   Steelypips, illustrious, industrious, and they hadn't a care: no spats
   in their vats, no rules, no schools, no gloom, no evil influence of the
   moon, no trouble from matter or antimatter-for they had a machine, a
   dream of a machine, with springs and gears and perfect in every respect.
   Penalty: 199
   ```

   With maximum line length 25, the output should look like:

   ```
   Not far from here, by
   a white sun, behind a
   green star, lived the
   Steelypips, illustrious,
   industrious, and they
   ```

[1] This is a line from Stanislaw Lem's *The Cyberiad*.

```
hadn't a care: no spats
in their vats, no rules,
no schools, no gloom,
no evil influence of the
moon, no trouble from
matter or antimatter-for
they had a machine, a
dream of a machine, with
springs and gears and
perfect in every respect.
Penalty: 137
```

2. **Subset Sum.** In this problem, you are given an integer *weight threshold* $W > 0$ and a list of n items $\{1, \ldots, n\}$ each with nonnegative weight w_i. Your task is to output a subset of items $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} w_i$ is as large as possible, subject to $\sum_{i \in S} w_i \leq W$.

   Design an analyze a dynamic program to solve **Subset Sum**. Your algorithm should run in $O(nW)$ time.

3. **Flow variant.** In the standard flow problem, we get an input $G = (V, E)$ a directed graph and edge capacities $c_e \geq 0$ limiting how much flow can pass along an edge. Consider the following two variants of the maximum flow problem.

   (a) It might be that each junction where water pipes meet is limited in how much water it can handle (no matter how much the pipes can carry). In this case, we want to add *vertex* capacities to our problem. The input is a directed G (with source s and sink $t \in V$), edge capacities $c_e \geq 0$, and vertex capacities $c_v \geq 0$ describing the upper limit of flow which can pass through that vertex. Give a polynomial-time algorithm to find the maximum $s \rightsquigarrow t$ flow in a network with both edge and vertex capacities.

   (b) It might be that there are multiple sources and multiple sinks in our flow network. In this case, the input is a directed G, a list of sources $\{s_1, \ldots, s_x\} \subset V$, a list of sinks $\{t_1, \ldots, t_y\} \subset V$, and edge capacities $c_e \geq 0$.

   Give a polynomial-time algorithm to find the maximum flow in a network with multiple sources and multiple sinks.

4. **Advertising contracts** (K& T 7.16)

   Back in the euphoric early days of the Web, people liked to claim that much of the enormous potential in a company like Yahoo! was in the "eyeballs"—the simple fact that millions of people look at its pages every day. Further, by convincing people to register personal data with the site, a site like Yahoo! can show each user an extremely targeted advertisement whenever the user visits the site, in a way that TV networks or magazines couldn't hope to match. So if a user has told Yahoo! that she is a 21-year-old computer science major at Swarthmore, the site can present a banner ad for apartments in Philadelphia suburbs; on the other hand, if she is a 50-year-old investment banker from Greenwich, CT, the site can display a banner ad pitching luxury cars instead.

But deciding on which ads to show to which people involves some serious computation behind the scenes. Suppose that the managers of a popular site have identified k distinct *demographic groups* $G_1, G_2, \ldots, G_k$. (Some may overlap.) The site has contracts with m different advertisers to show a certain number of copies of their ads to users of the site. Here's what a contract with the i^{th} advertiser looks like:

- For a subset $X_i \subseteq \{G_1, \ldots, G_k\}$ of the demographic groups, advertiser i wants ads shown only to users who belong to at least one of the groups listed in X_i.
- For a number r_i, advertiser i want its ads shown to at least r_i users each minute.

Consider the problem of designing a good *advertising policy* — a way to show a single ad to each user of the site. (Imagine a world where each user saw only *one* ad per site.) Suppose at a given minute, there are n users visiting the site. Because we have registration about each of the users, we know that user j belongs to a subset U_j of the demographic groups.

The problem is: is there a way to show a single ad to each user so that the site's contracts with each of the m advertisers is satisfied for this minute?

Give an efficient algorithm to decide if this is possible, and if so, to actually choose an ad to show each user.