--- Day 7: No Space Left On Device ---

You can hear birds chirping and raindrops hitting leaves as the expedition proceeds. Occasionally, you can even hear much louder sounds in the distance; how big do the animals get out here, anyway?

The device the Elves gave you has problems with more than just its communication system. You try to run a system update:

```
$ system-update --please --pretty-please-with-sugar-on-top
Error: No space left on device
```

Perhaps you can delete some files to make space for the update?

You browse around the filesystem to assess the situation and save the resulting terminal output (your puzzle input). For example:

```
$ cd /
$ ls
dir a
14848514 b.txt
8504156 c.dat
dir d
$ cd a
$ ls
dir e
29116 f
2557 g
62596 h.lst
$ cd e
$ ls
584 i
$ cd ..
$ cd ..
$ cd d
$ ls
4060174 j
8033020 d.log
5626152 d.ext
7214296 k
```

The filesystem consists of a tree of files (plain data) and directories (which can contain other directories or files). The outermost directory is called `/`. You can navigate around the filesystem, moving into or out of directories and listing the contents of the directory you're currently in.

Within the terminal output, lines that begin with `$` are *commands you executed*, very much like some modern computers:

- `cd` means *change directory*. This changes which directory is the current directory, but the specific result depends on the argument:
  - `cd x` moves *in* one level: it looks in the current directory for the directory named `x` and makes it the current directory.
  - `cd ..` moves *out* one level: it finds the directory that contains the current directory, then makes that directory the current directory.
  - `cd /` switches the current directory to the outermost directory, `/`.
- `ls` means *list*. It prints out all of the files and directories immediately contained by the current directory:
  - `123 abc` means that the current directory contains a file named `abc`with size `123`.
  - `dir xyz` means that the current directory contains a directory named `xyz`.

Given the commands and output in the example above, you can determine that the filesystem looks visually like this:

```
- / (dir)
  - a (dir)
    - e (dir)
      - i (file, size=584)
    - f (file, size=29116)
    - g (file, size=2557)
    - h.lst (file, size=62596)
  - b.txt (file, size=14848514)
  - c.dat (file, size=8504156)
  - d (dir)
    - j (file, size=4060174)
    - d.log (file, size=8033020)
    - d.ext (file, size=5626152)
    - k (file, size=7214296)
```

Here, there are four directories: `/` (the outermost directory), `a` and `d` (which are in `/`), and `e` (which is in `a`). These directories also contain files of various sizes.

Since the disk is full, your first step should probably be to find directories that are good candidates for deletion. To do this, you need to determine the *total size*of each directory. The total size of a directory is the sum of the sizes of the files it contains, directly or indirectly. (Directories themselves do not count as having any intrinsic size.)

The total sizes of the directories above can be found as follows:

- The total size of directory `e` is *584* because it contains a single file `i` of size 584 and no other directories.
- The directory `a` has total size *94853* because it contains files `f` (size 29116), `g` (size 2557), and `h.lst` (size 62596), plus file `i` indirectly (`a`contains `e` which contains `i`).
- Directory `d` has total size *24933642*.
- As the outermost directory, `/` contains every file. Its total size is *48381165*, the sum of the size of every file.

To begin, find all of the directories with a total size of *at most 100000*, then calculate the sum of their total sizes. In the example above, these directories are `a` and `e`; the sum of their total sizes is *`95437`* (94853 + 584). (As in this example, this process can count files more than once!)

Find all of the directories with a total size of at most 100000. *What is the sum of the total sizes of those directories?*

Your puzzle answer was `1443806`.

--- Part Two ---

Now, you're ready to choose a directory to delete.

The total disk space available to the filesystem is *`70000000`*. To run the update, you need unused space of at least *`30000000`*. You need to find a directory you can delete that will *free up enough space* to run the update.

In the example above, the total size of the outermost directory (and thus the total amount of used space) is `48381165`; this means that the size of the *unused* space must currently be `21618835`, which isn't quite the `30000000`required by the update. Therefore, the update still requires a directory with total size of at least `8381165` to be deleted before it can run.

To achieve this, you have the following options:

- Delete directory `e`, which would increase unused space by `584`.
- Delete directory `a`, which would increase unused space by `94853`.
- Delete directory `d`, which would increase unused space by `24933642`.
- Delete directory `/`, which would increase unused space by `48381165`.

Directories `e` and `a` are both too small; deleting them would not free up enough space. However, directories `d` and `/` are both big enough! Between these, choose the *smallest*: `d`, increasing unused space by *`24933642`*.

Find the smallest directory that, if deleted, would free up enough space on the filesystem to run the update. *What is the total size of that directory?*

Your puzzle answer was `942298`.