CSE 250
Lecture 33

How to Organize Your Cat Pictures

I have [illegible] of cat pictures

I want to organize my cat pictures

Serenity Pictures

 Funny Bonnet.jpg

 Rocker.jpg

 MrsClawsCrossedMeForTheLastTime.jpg

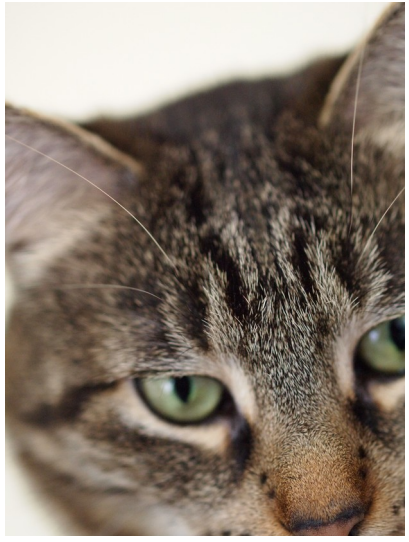
Juliet Pictures

 ComfyLaundry.jpg

 Closeup.jpg

 Sunbeam.jpg

What just happened?

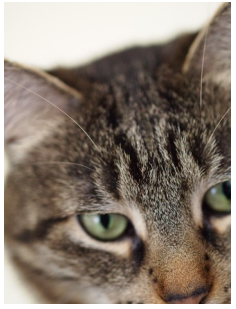


=

```
0000000 | d8ff e0ff 1000 464a 4649 0100 0101 3a01
0000016 | 3a01 0000 e1ff 002e 7845 6669 0000 4949
0000032 | 002a 0008 0000 000d 010e 0002 0084 0000
0000048 | 00aa 0000 010f 0002 0018 0000 012e 0000
0000064 | 0110 0002 0011 0000 0146 0000 0112 0003
0000080 | 0001 0000 0001 0000 011a 0005 0001 0000
0000096 | 0158 0000 011b 0005 0001 0000 0160 0000
0000112 | 0128 0003 0001 0000 0002 0000 0131 0002
0000128 | 000d 0000 0168 0000 0132 0002 0014 0000
0000144 | 0176 0000 0213 0003 0001 0000 0002 0000
0000160 | 8769 0004 0001 0000 039a 0000 8825 0004
          .... .... .... .... .... .... .... ....
```

What just happened?

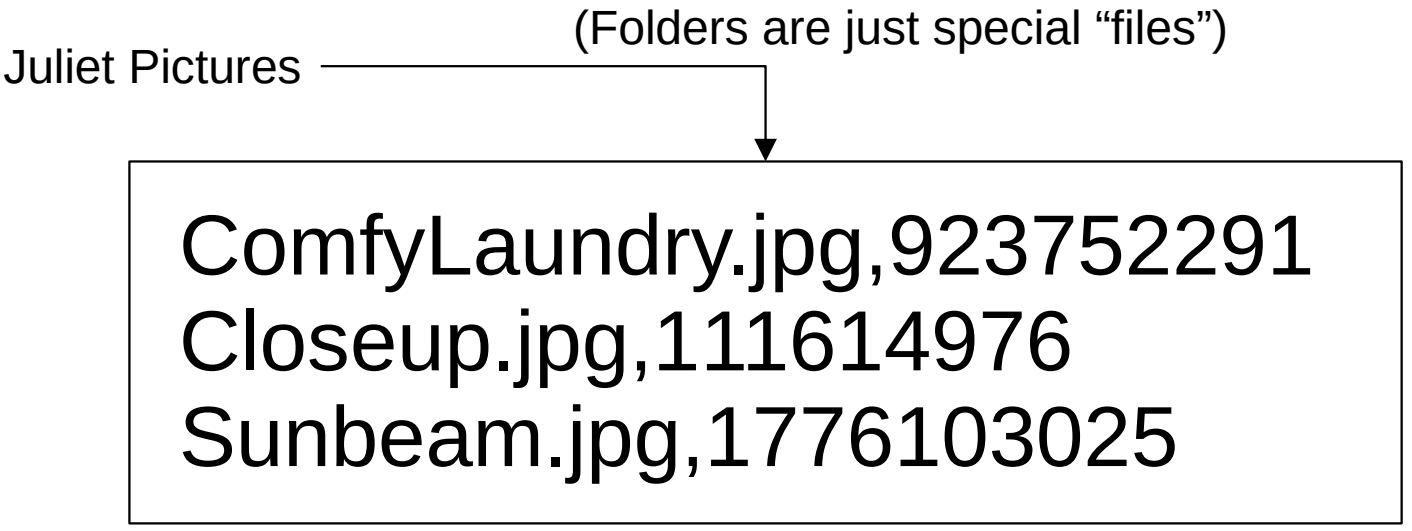
Byte 111614976
(start of cat picture)



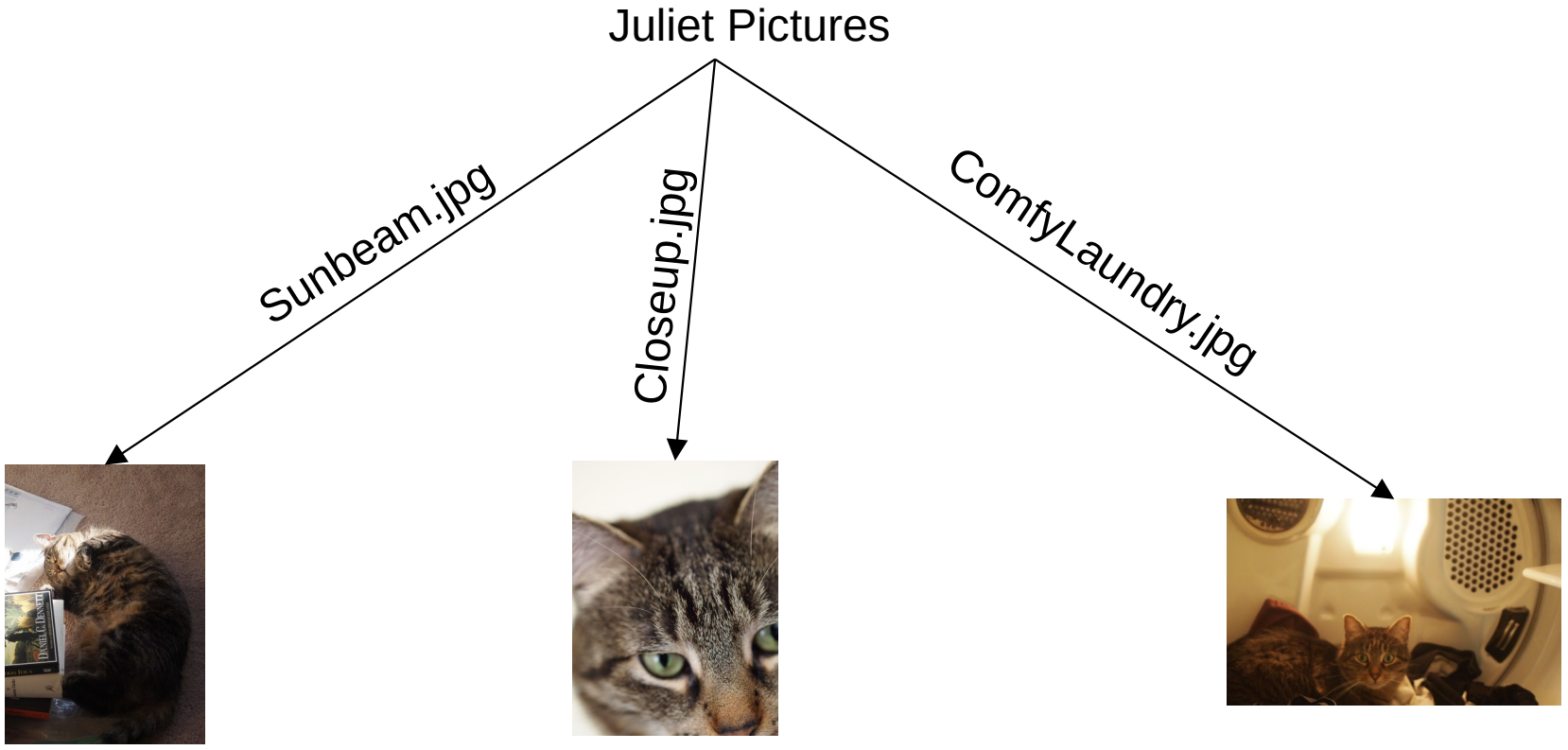
111614944	1bf7	a1f1	5c20	b3d6	c0d9	5262	4121	1973
111614960	7a65	e6b2	7488	cad4	a360	3a06	5c02	4b55
111614976	**d8ff**	**e0ff**	**1000**	**464a**	**4649**	**0100**	**0101**	**3a01**
111614992	**3a01**	**0000**	**e1ff**	**002e**	**7845**	**6669**	**0000**	**4949**
111615008	**002a**	**0008**	**0000**	**000d**	**010e**	**0002**	**0084**	**0000**
111615024	**00aa**	**0000**	**010f**	**0002**	**0018**	**0000**	**012e**	**0000**
111615040	**0110**	**0002**	**0011**	**0000**	**0146**	**0000**	**0112**	**0003**
111615056	**0001**	**0000**	**0001**	**0000**	**011a**	**0005**	**0001**	**0000**
111615072	**0158**	**0000**	**011b**	**0005**	**0001**	**0000**	**0160**	**0000**
111615088	**0128**	**0003**	**0001**	**0000**	**0002**	**0000**	**0131**	**0002**
111615104	**000d**	**0000**	**0168**	**0000**	**0132**	**0002**	**0014**	**0000**
111615120	**0176**	**0000**	**0213**	**0003**	**0001**	**0000**	**0002**	**0000**
111615136	**8769**	**0004**	**0001**	**0000**	**039a**	**0000**	**8825**	**0004**
	....	....	....	....	....	....	....	....

=

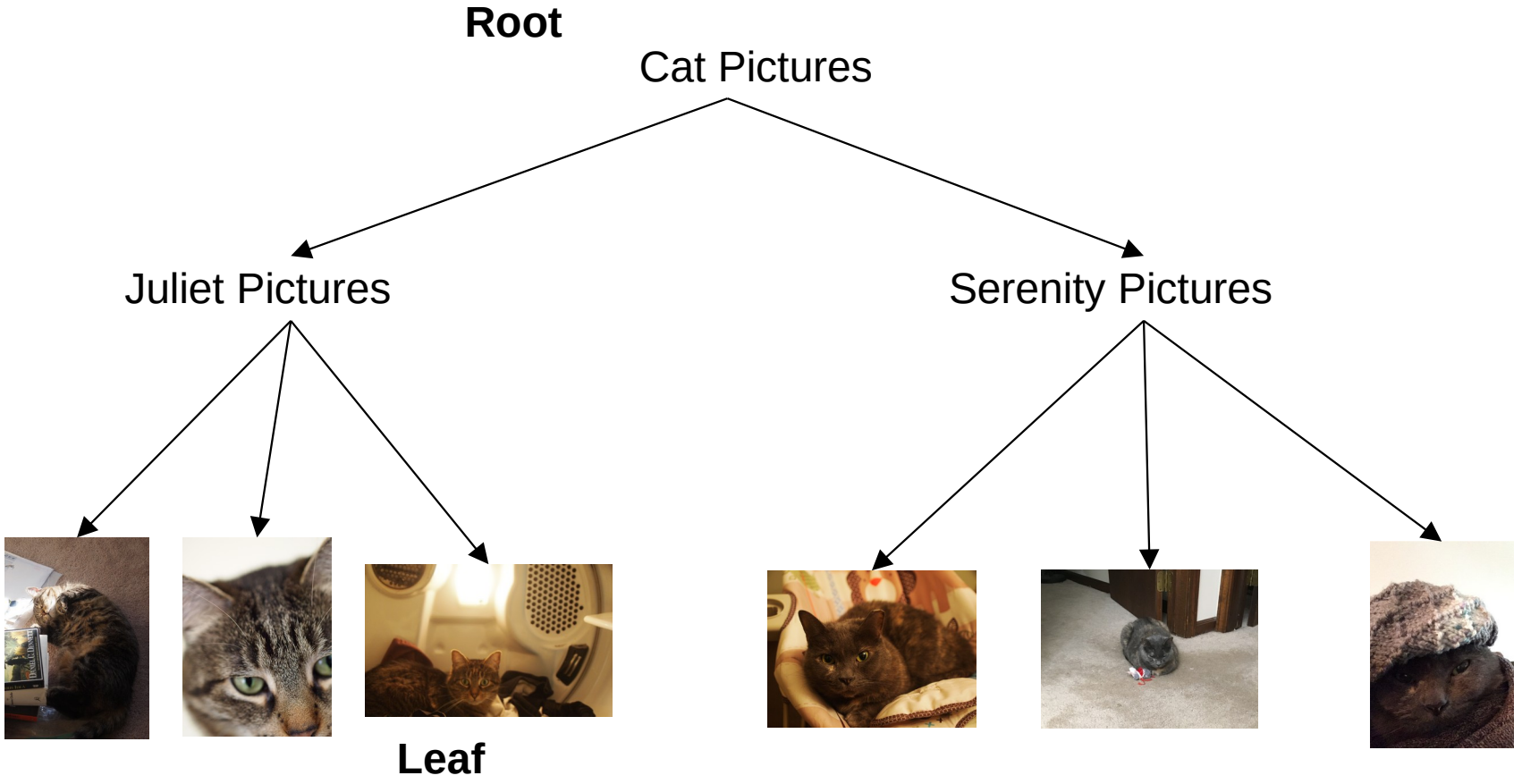
What just happened?



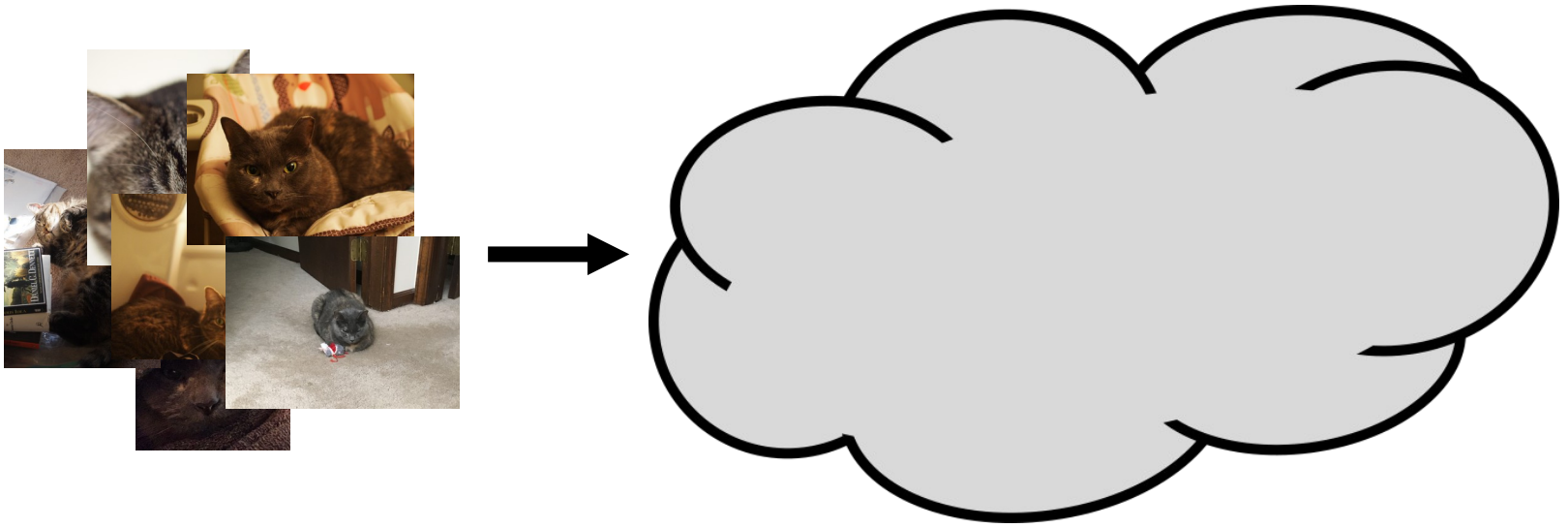
It’s a tree



It’s a tree

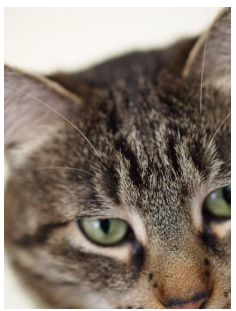


Off to the Internet



Off to the Internet

Byte 11614976
(start of cat picture)

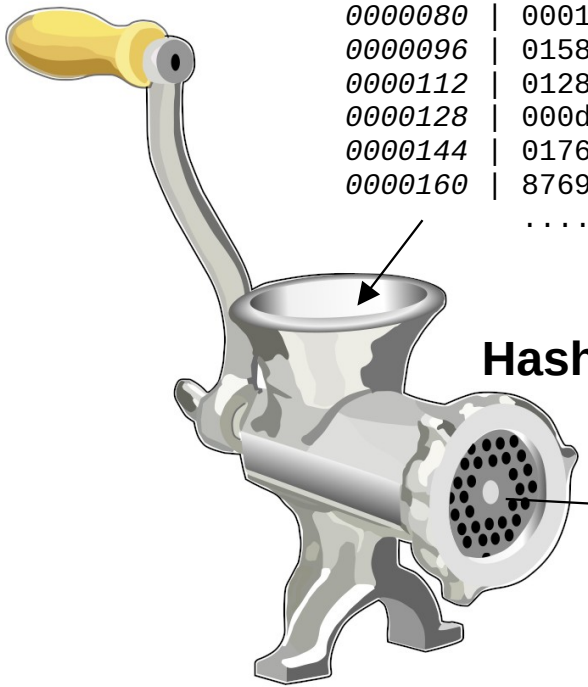


```
11614944 | 1bf7 a1f1 5c20 b3d6 c0d9 5262 4121 1973
11614960 | 7a65 e6b2 7488 cad4 a360 3a06 5c02 4b53
11614976 | d8ff e0ff 1000 464a 4649 0100 0101 3a01
11614992 | 3a01 0000 e1ff 002e 7845 6669 0000 4949
11615008 | 002a 0008 0000 000d 010e 0002 0084 0000
11615024 | 00aa 0003 010f 0002 0018 0000 012e 0000
11615040 | 0110 0002 0011 0000 0146 0000 0112 0003
11615056 | 0001 0000 0001 0000 011a 0005 0001 0000
11615072 | 0158 0000 011b 0005 0001 0000 0160 0000
11615088 | 0128 0003 0001 0000 0002 0000 0131 0002
11615104 | 000d 0000 0168 0000 0132 0002 0014 0000
11615120 | 0176 0000 0213 0003 0001 0000 0002 0000
11615136 | 8769 0004 0001 0000 039a 0000 8825 0004
         | .... .... .... .... .... .... .... ....
```

=

Off to the internet

```
0000000 | d8ff e0ff 1000 464a 4649 0100 0101 3a01
0000016 | 3a01 0000 e1ff 002e 7845 6669 0000 4949
0000032 | 002a 0008 0000 000d 010e 0002 0084 0000
0000048 | 00aa 0000 010f 0002 0018 0000 012e 0000
0000064 | 0110 0002 0011 0000 0146 0000 0112 0003
0000080 | 0001 0000 0001 0000 011a 0005 0001 0000
0000096 | 0158 0000 011b 0005 0001 0000 0160 0000
0000112 | 0128 0003 0001 0000 0002 0000 0131 0002
0000128 | 000d 0000 0168 0000 0132 0002 0014 0000
0000144 | 0176 0000 0213 0003 0001 0000 0002 0000
0000160 | 8769 0004 0001 0000 039a 0000 8825 0004
          .... .... .... .... .... .... .... ....
```

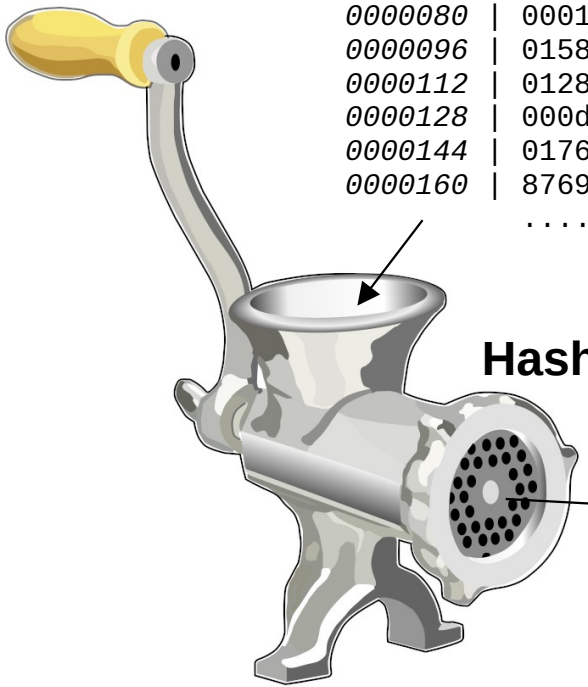


Hash Function

```
4fbabf25ba16e64b7dbc7632fefa7fb6b39edfc52b04a4c3e10515af21b9d7ee
```

Off to the internet

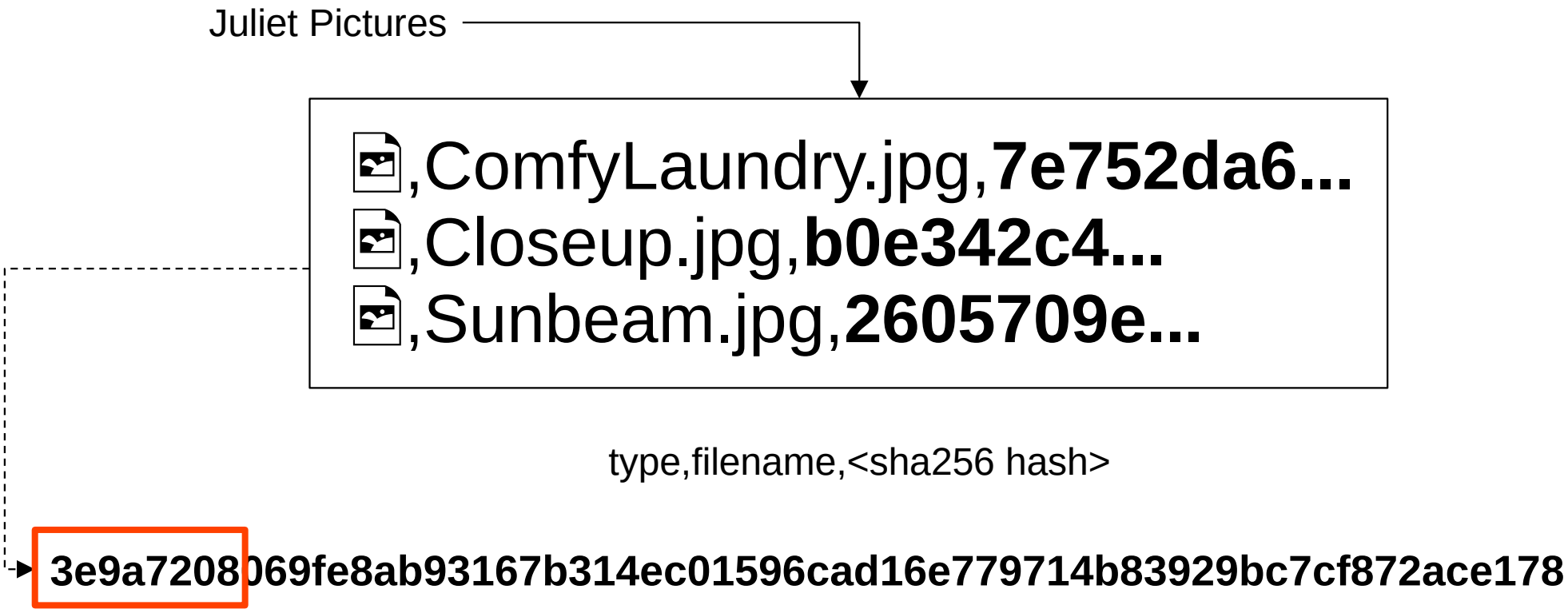
```
0000000 | d8ff e0ff 1000 464a 4649 0100 0101 3a01
0000016 | 3a01 0000 e1ff 002e 7845 6669 0000 4949
0000032 | 002a 0008 0010 000d 010e 0002 0084 0000
0000048 | 00aa 0000 010f 0002 0018 0000 012e 0000
0000064 | 0110 0002 0011 0000 0146 0000 0112 0003
0000080 | 0001 0000 0001 0000 011a 0005 0001 0000
0000096 | 0158 0000 011b 0005 0001 0000 0160 0000
0000112 | 0128 0003 0001 0000 0002 0000 0131 0002
0000128 | 000d 0000 0168 0000 0132 0002 0014 0000
0000144 | 0176 0000 0213 0003 0001 0000 0002 0000
0000160 | 8769 0004 0001 0000 039a 0000 8825 0004
          .... .... .... .... .... .... .... ....
```



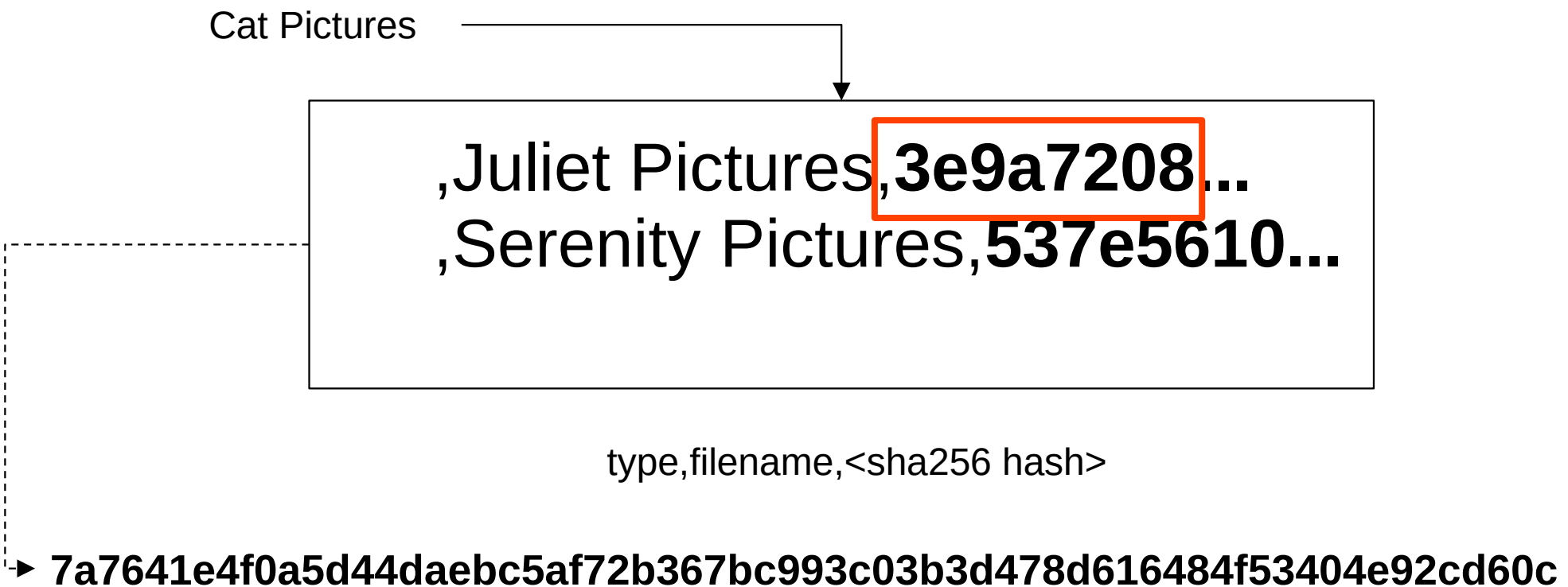
Hash Function

e552cc83a5e02421962eaa56c46542365c91e9476247db9605fa82ab4635a7d8

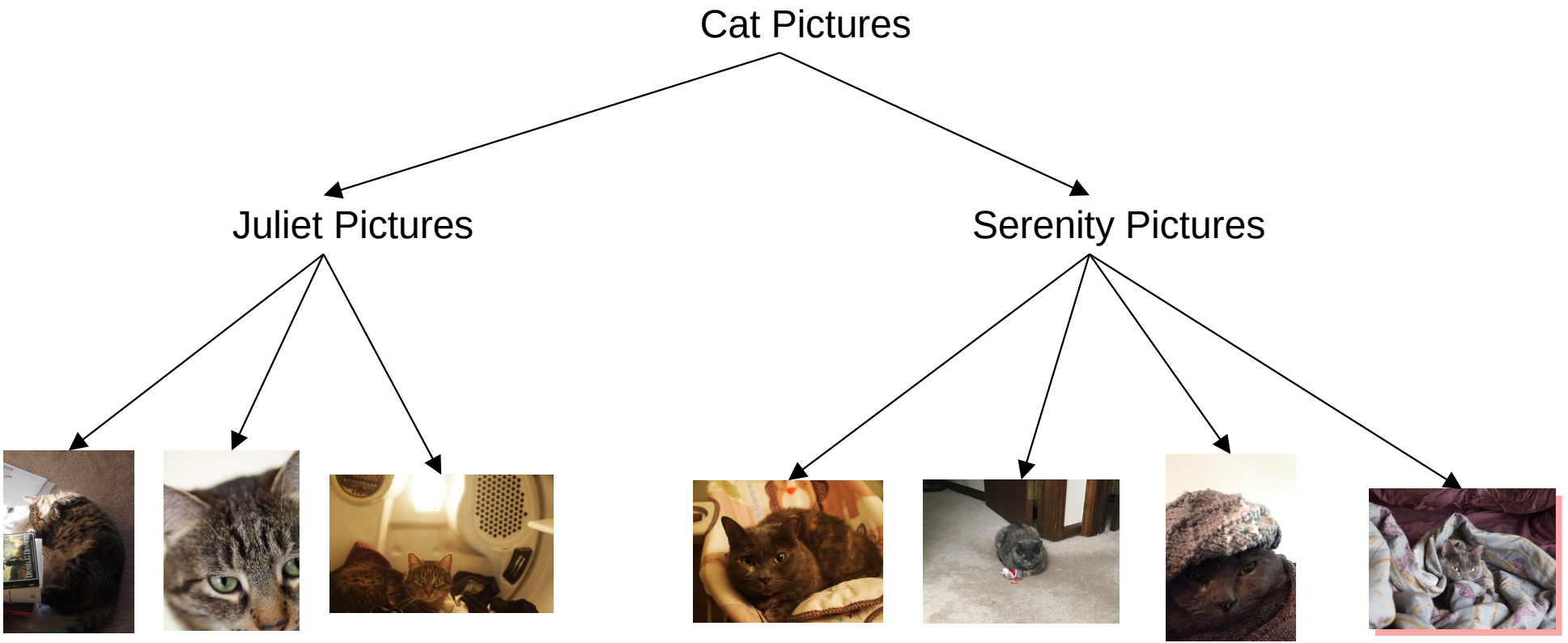
Off to the Internet



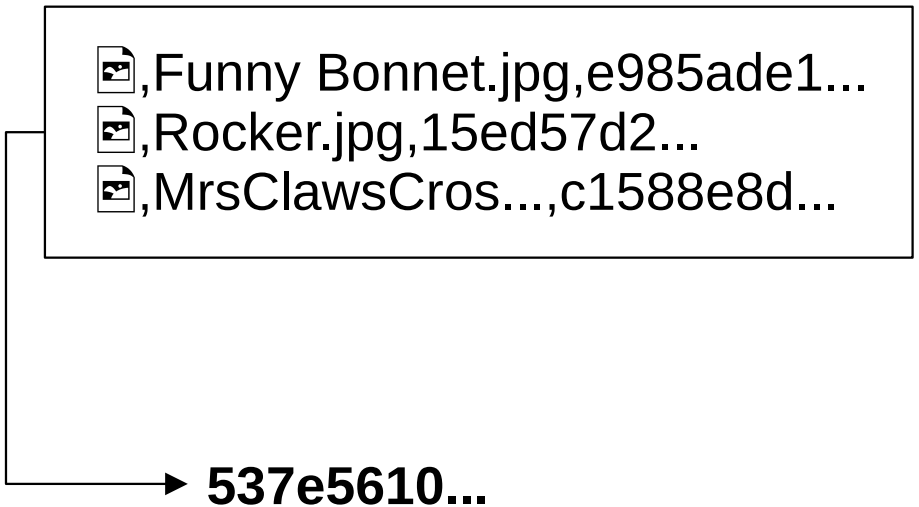
Off to the Internet

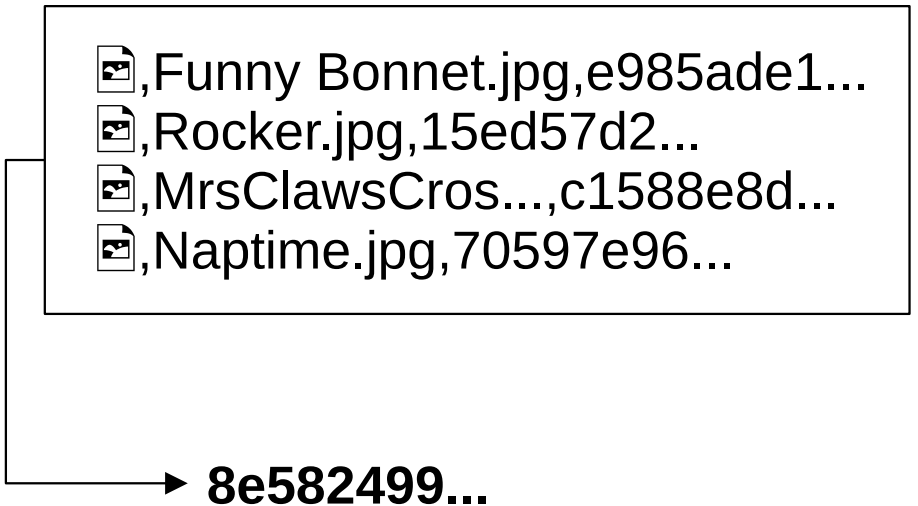


Changing data

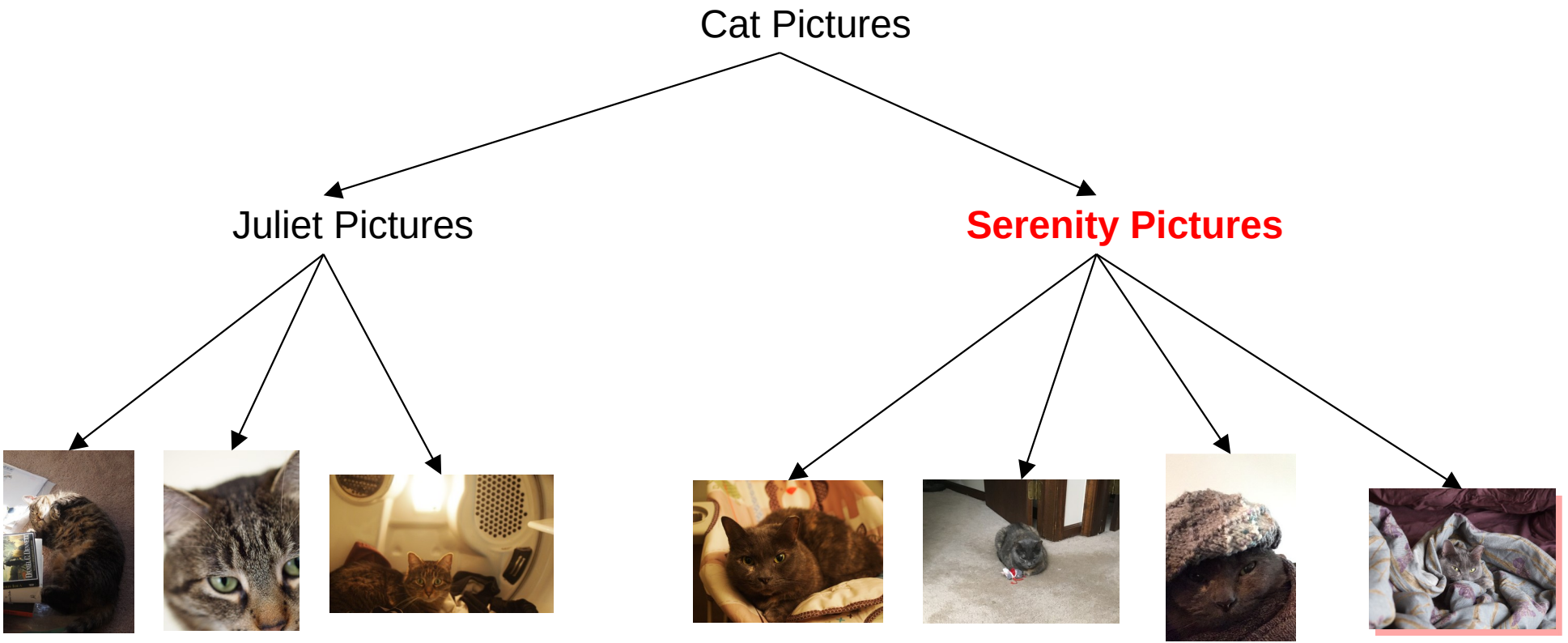


Changing data

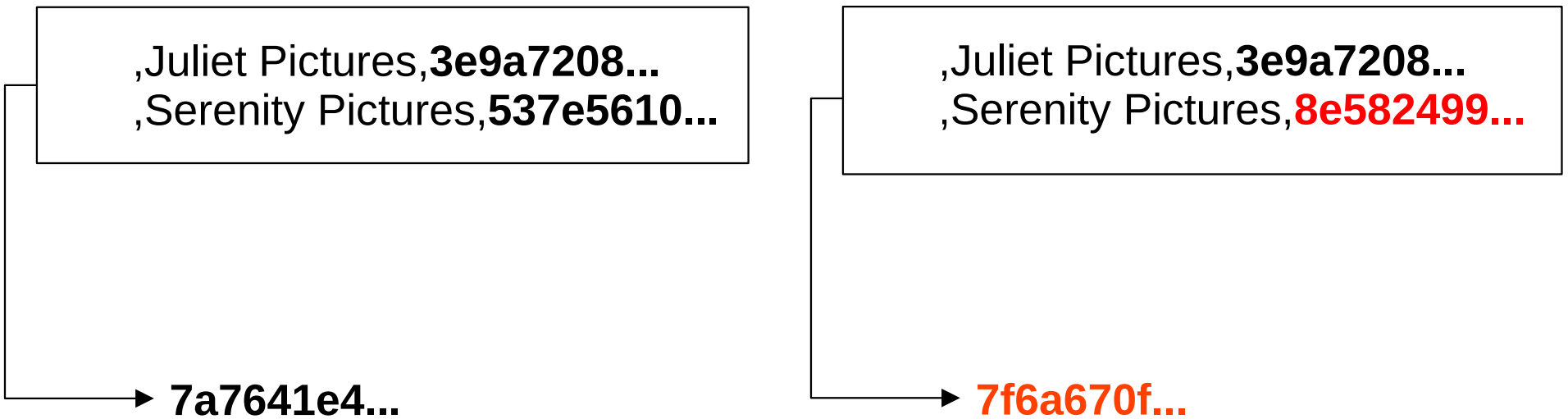




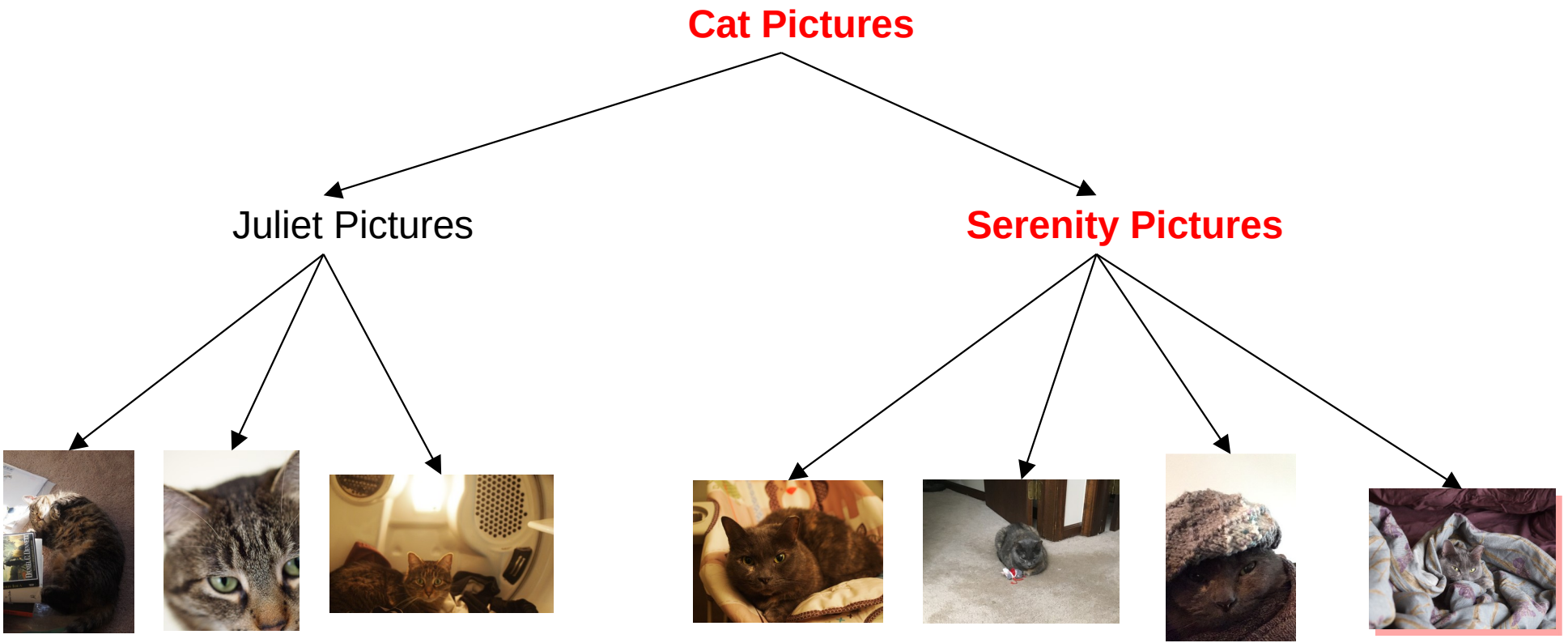
Changing data



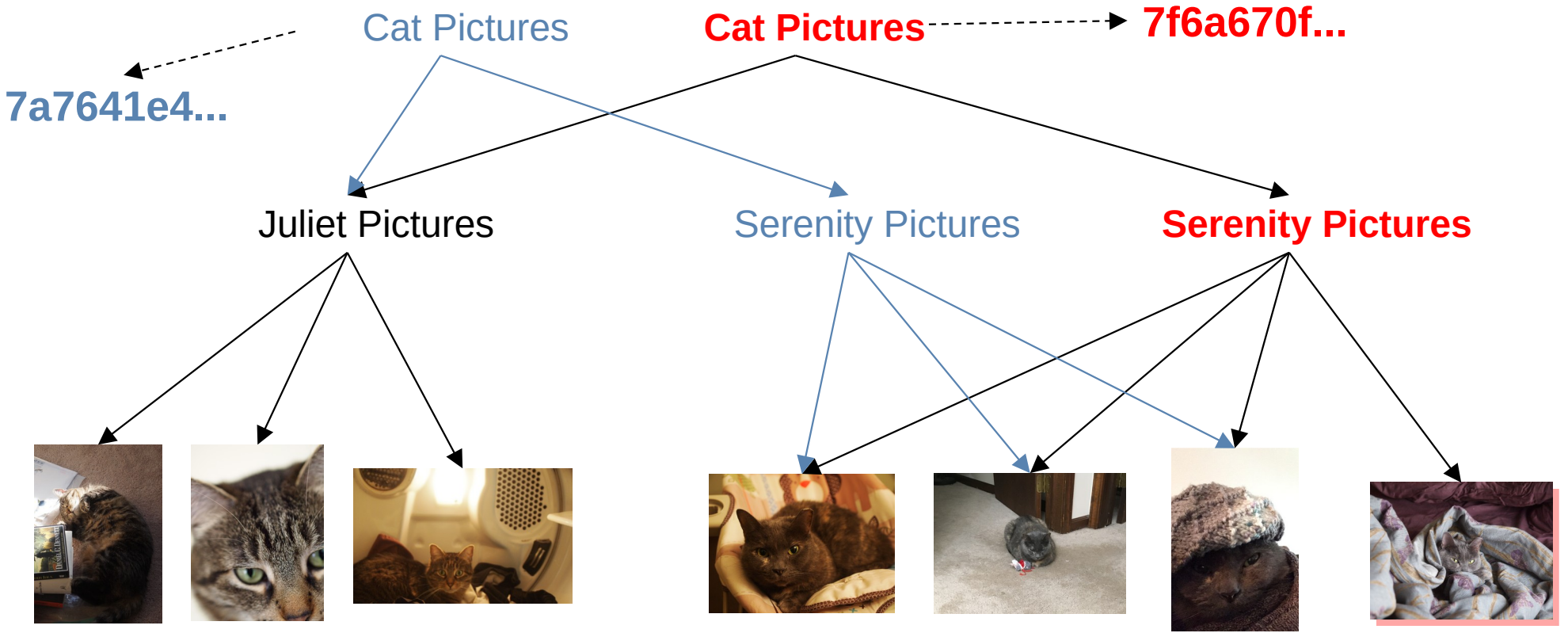
Changing data



Changing data



… but old “versions” don’t go away



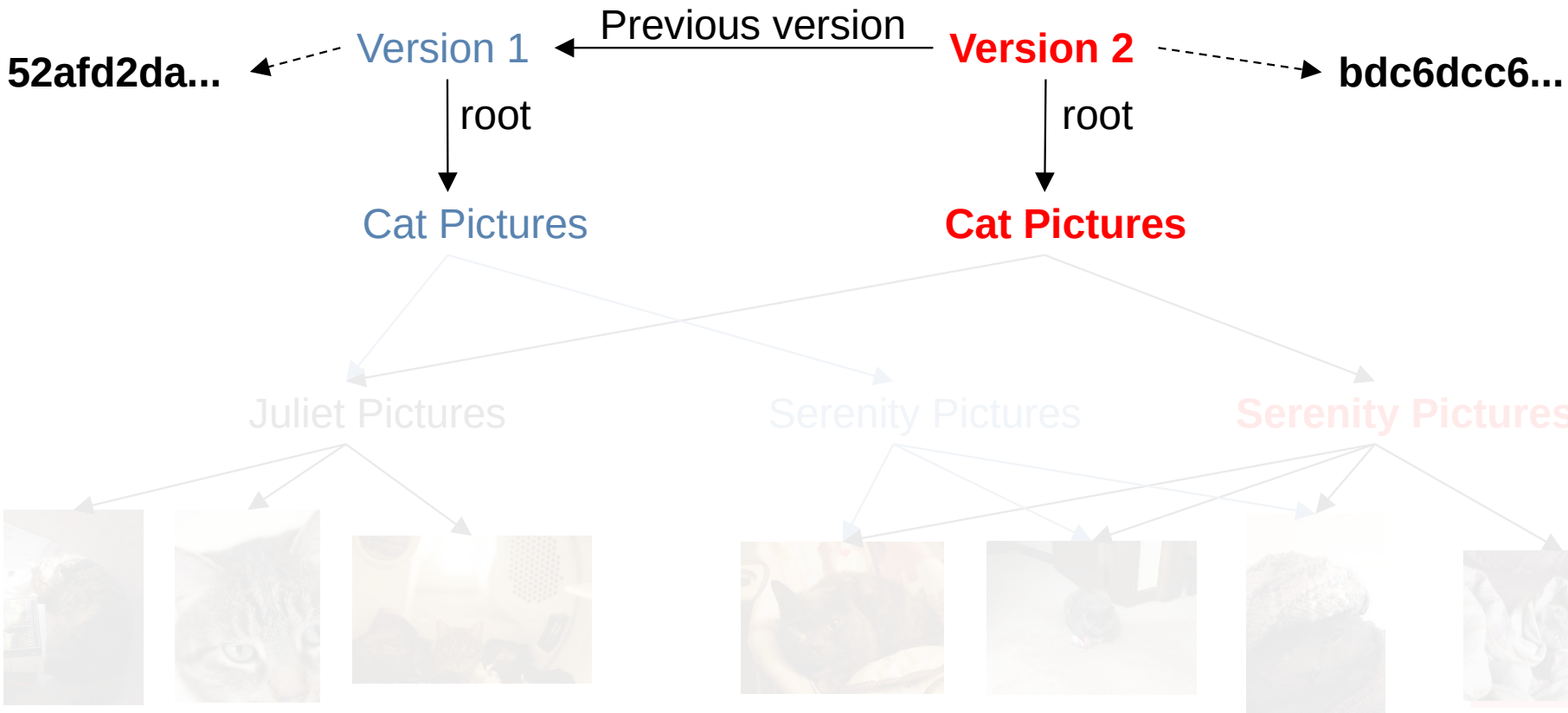
Where are my cats?

- The files aren’t a tree any more
  - It’s a directed graph
  - It doesn’t have cycles
  - (Directed Acyclic Graph; DAG)
- Too many roots; How do we keep track of them?
  - Add more vertices and edges to track version metadata

Where are my cats?

- Record information for each version:
  - ... the version's root
  - ... a reference to the previous version
  - ... what time the change happened
  - ... who made the change?

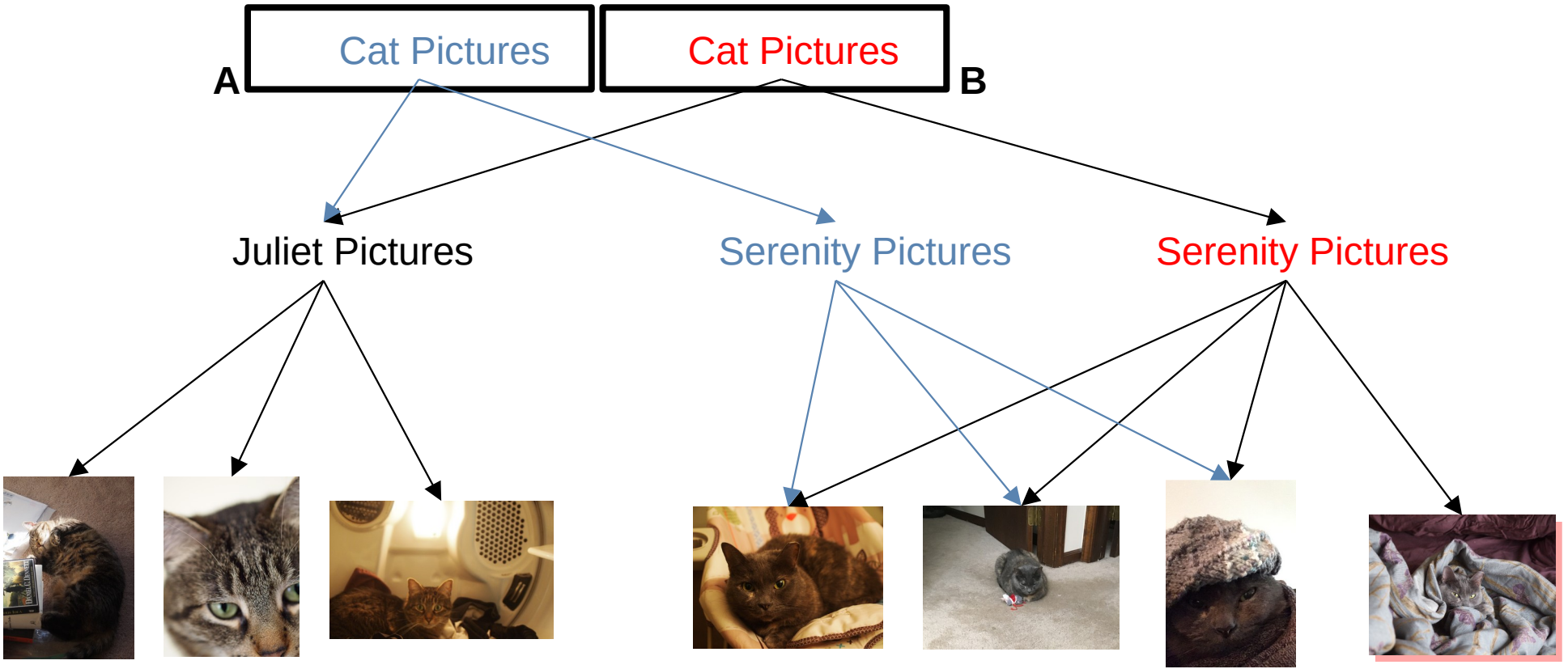
Versioned Cats



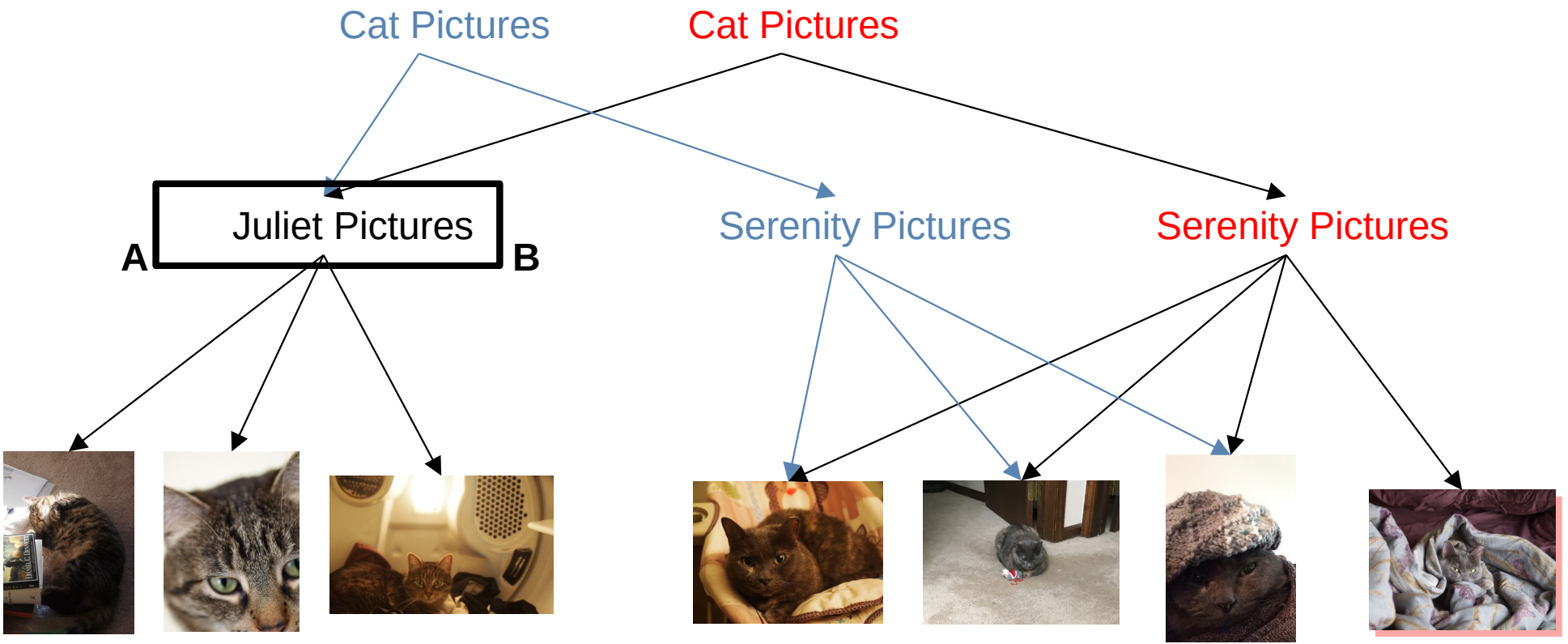
Multiple Versions

- What files changed between versions?
  - often called a “diff”
- When did a file appear?
- What versions modified a specific file?

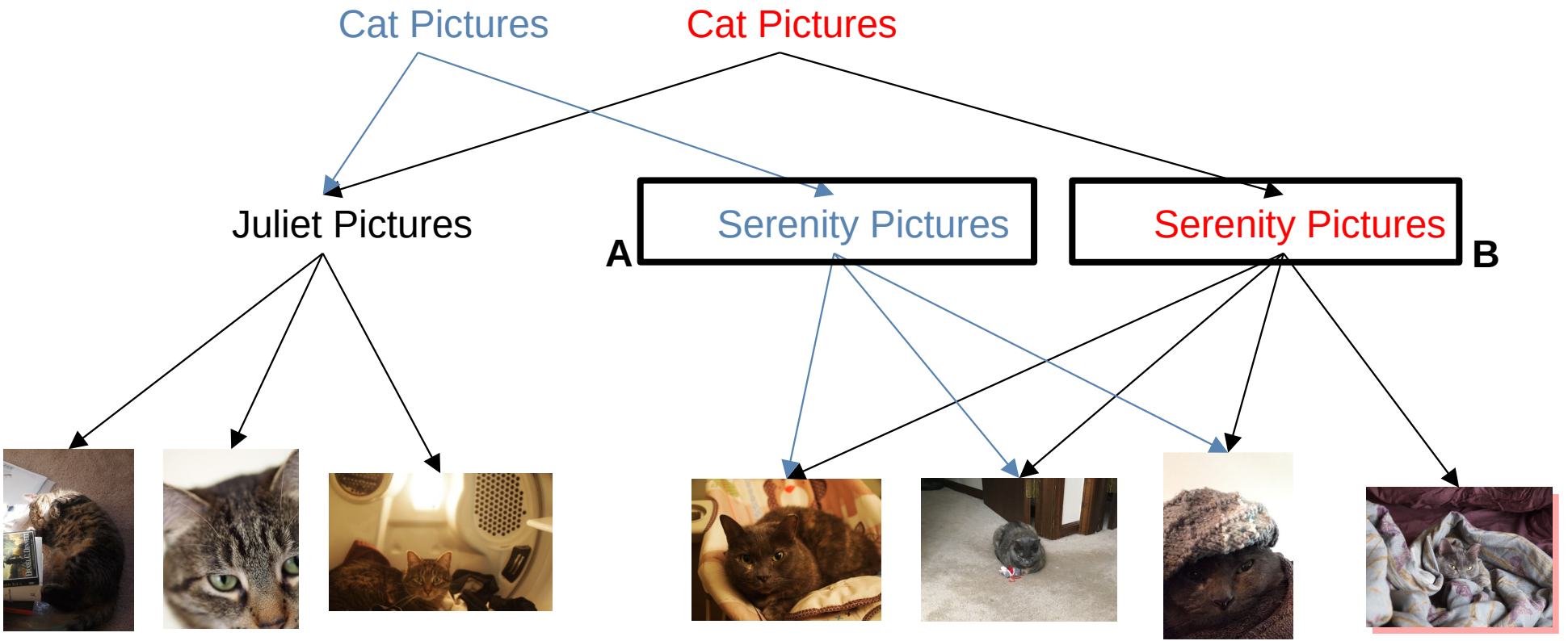
Computing a diff



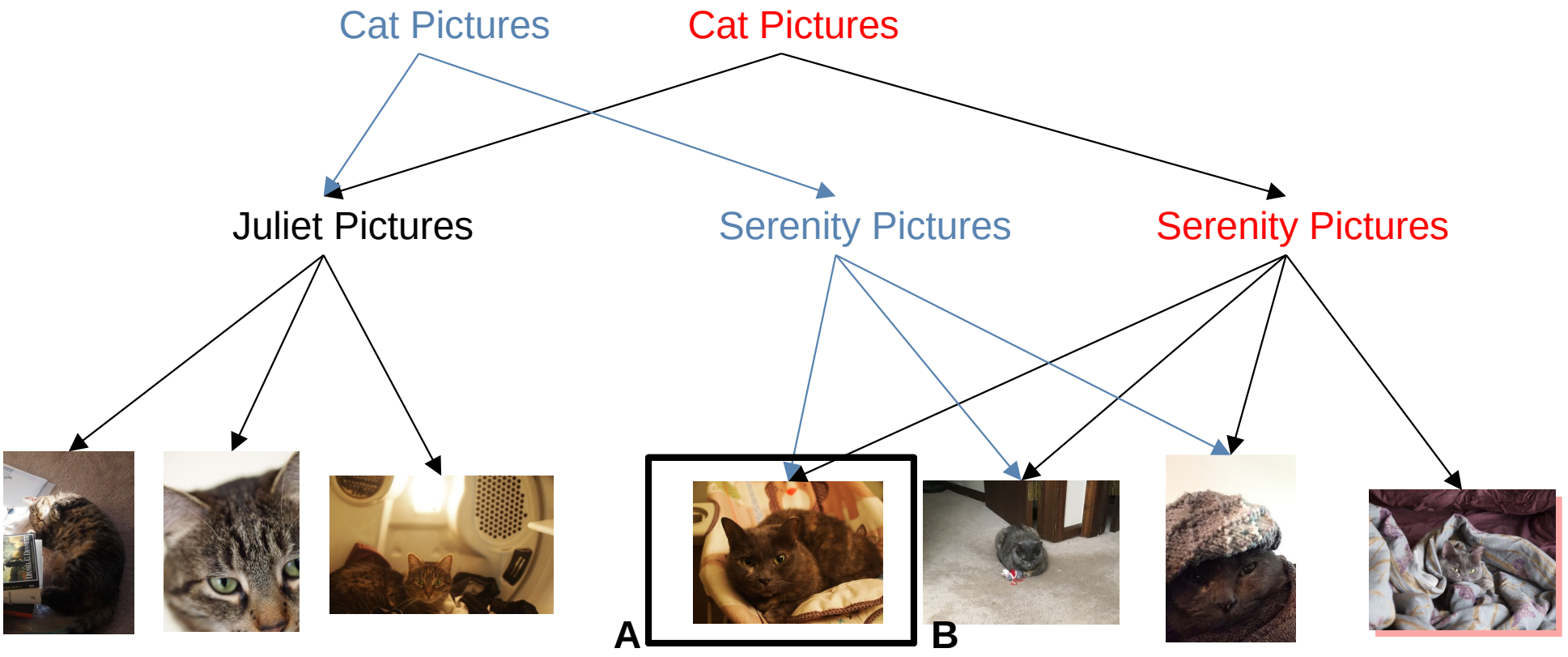
Computing a diff



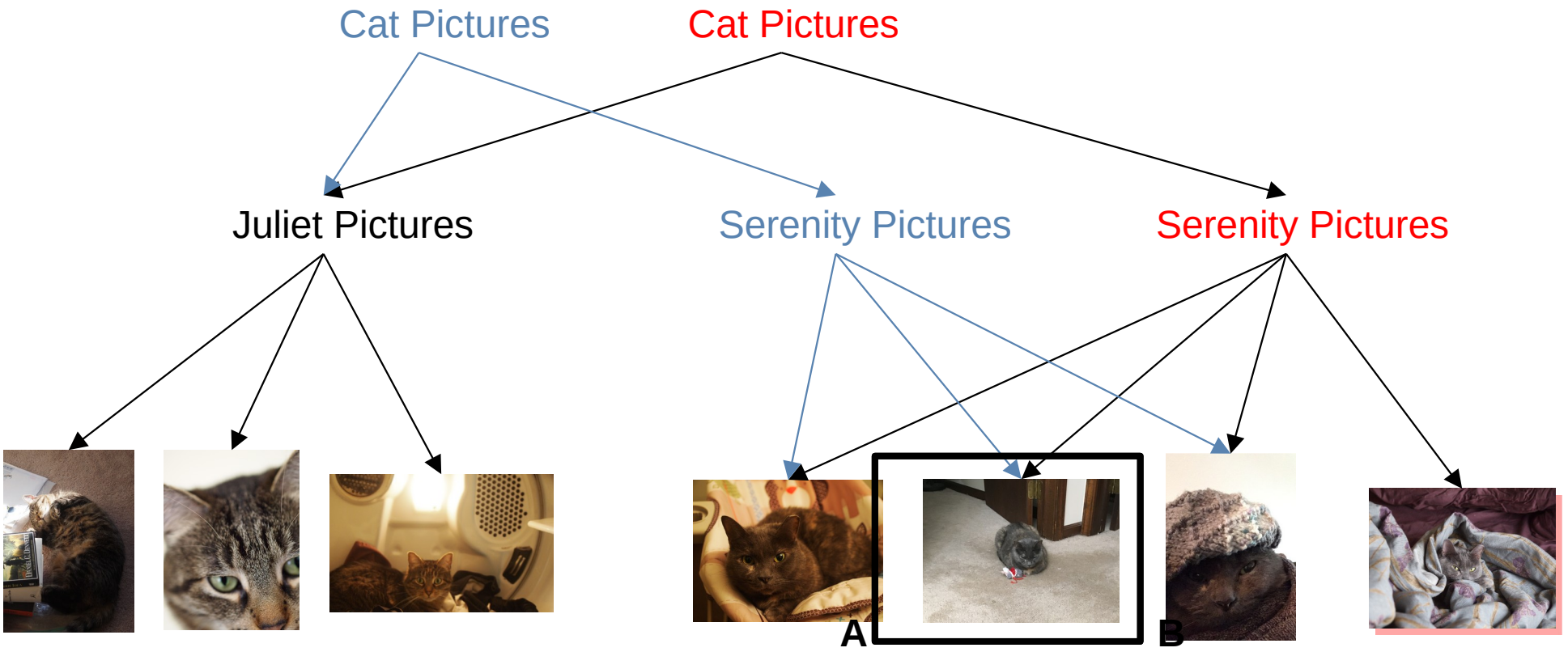
Computing a diff



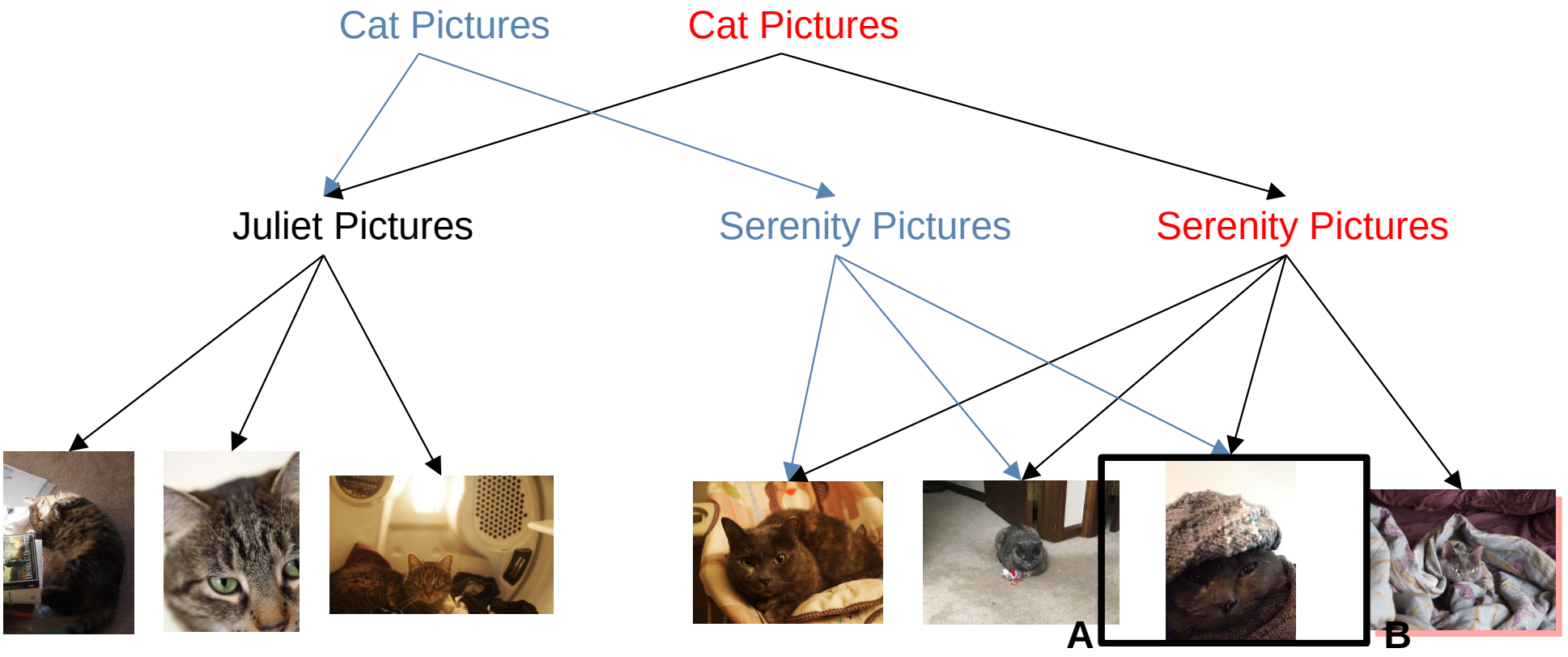
Computing a diff



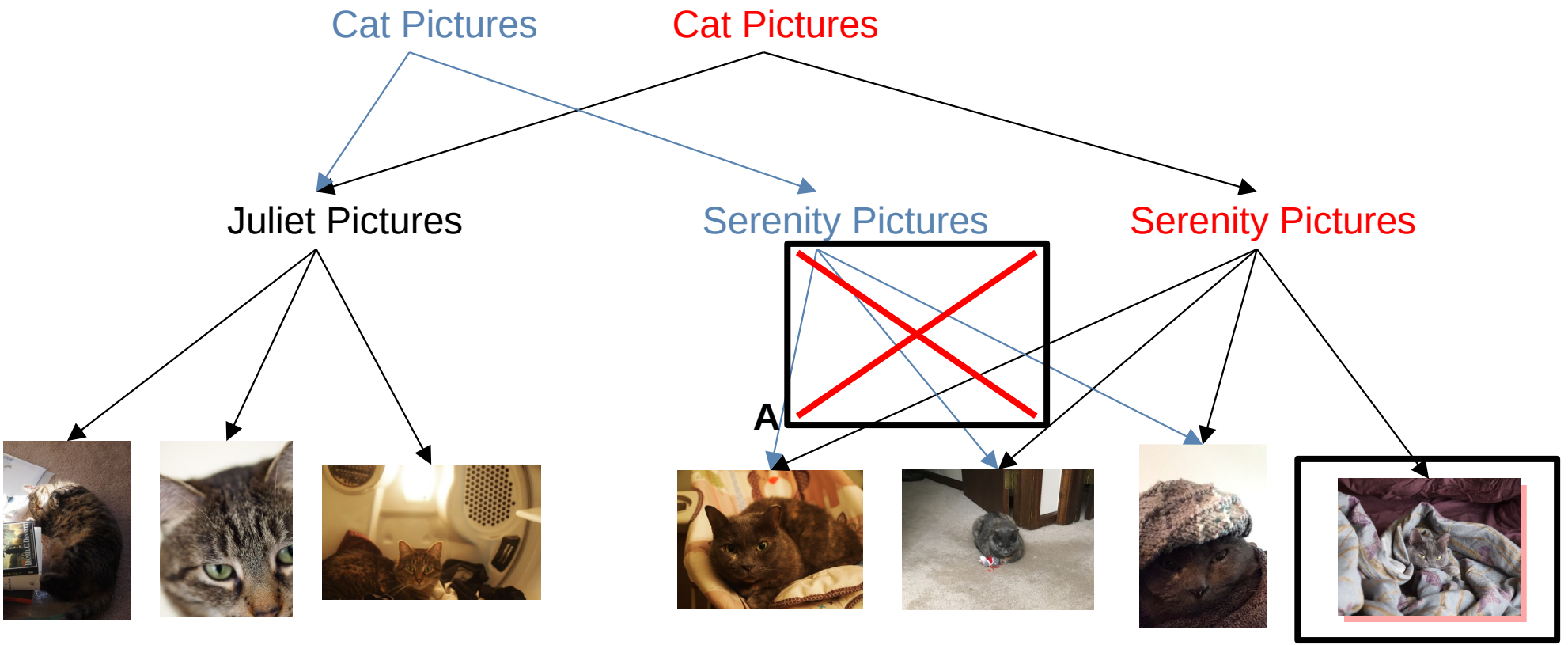
Computing a diff



Computing a diff



Computing a diff

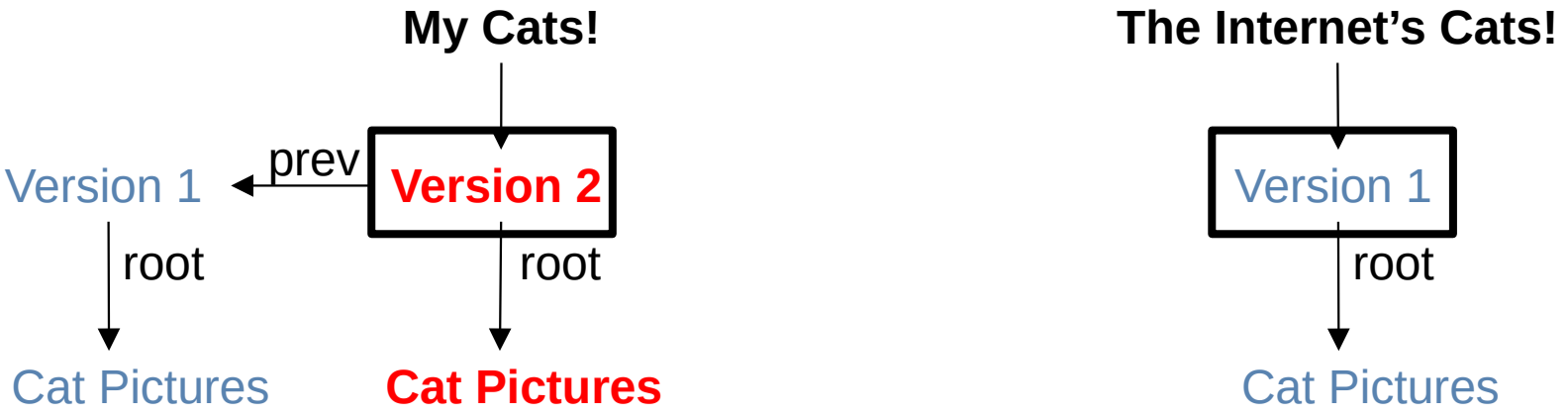


Computing a diff

```
def Diff(a: Vertex, b: Vertex, path: String, diff: ArrayBuffer[String]): Unit =
{
  if(a == b) { return } /* Trivial case, a and b are exactly the same; no diffs */
  for(edgeA <- a.outEdges){
    val edgeB = b.outEdges.find { _.label == edgeA.Label }
    if(edgeB.isDefined) {
      /* Recur */
      Diff(edgeA.dest,                                              ff)
    } else {
      diff.append(s“$p
    }
  }
  for(edgeB <- b.outEdges){
    if(a.outEdges.find { _.label == edgeB.Label }.isDefined){ /* already Diffed */ }
    else {
      diff.append(s“$path/${edgeB.label} is in B, but not A”)
    }
  }
}
```

This is DFS!

Pushing my cats onto the Internet

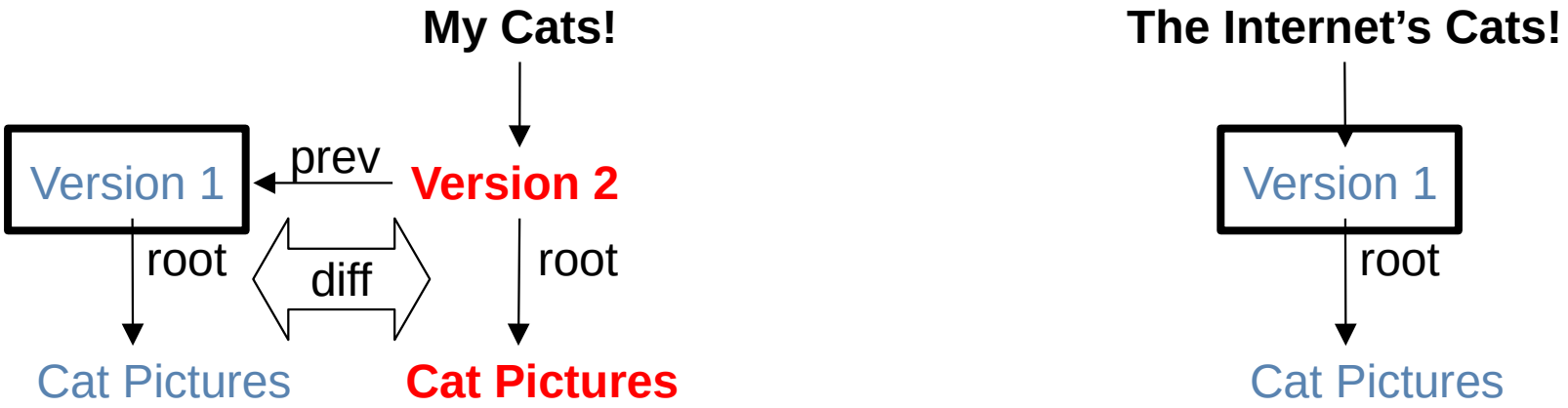


What do I need to send to the internet for it to have version 2?

"Version 2"
The new Cat Pictures
The new Serenity Pictures
The new picture

Why?

Pushing my cats onto the Internet

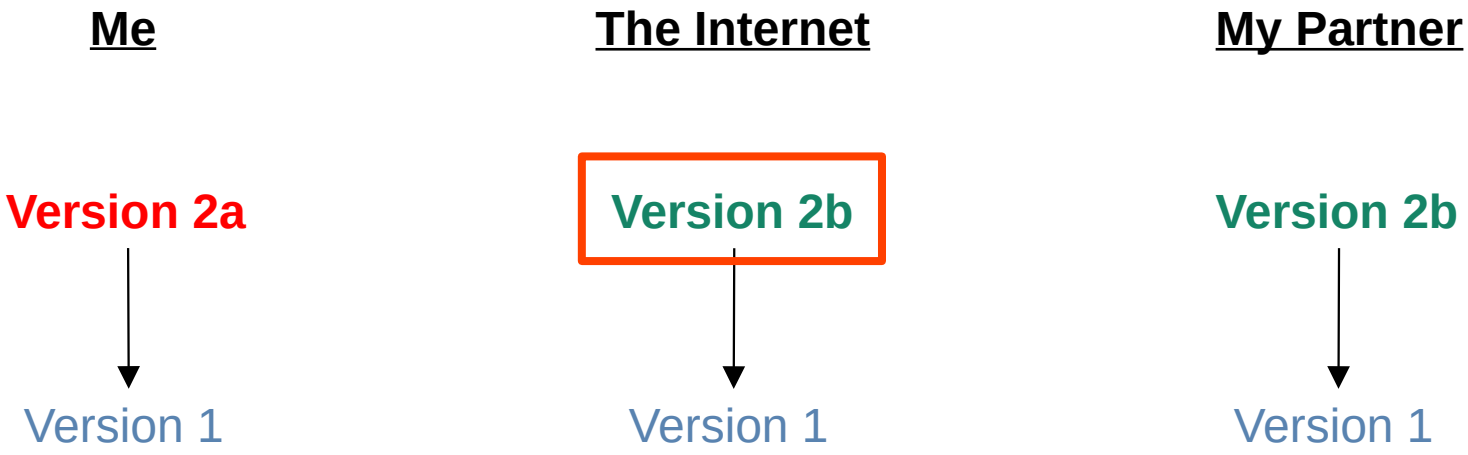


1. Find the internet's version in my version history
2. Send everything added in the new versions

Multi-user cats

Me	The Internet	My Partner
Version 2a	Version 1	**Version 2b**
↓		↓
Version 1		Version 1

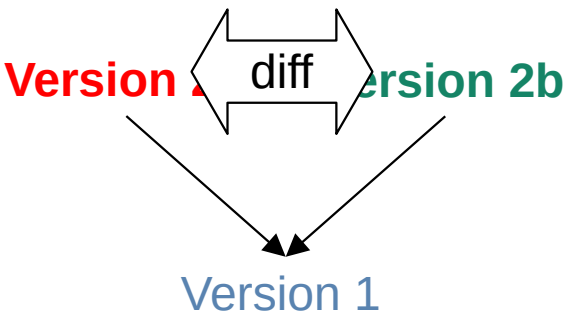
Multi-user cats



Version 2b is not in my history!

Multi-user cats

Me



The Internet

Version 2b

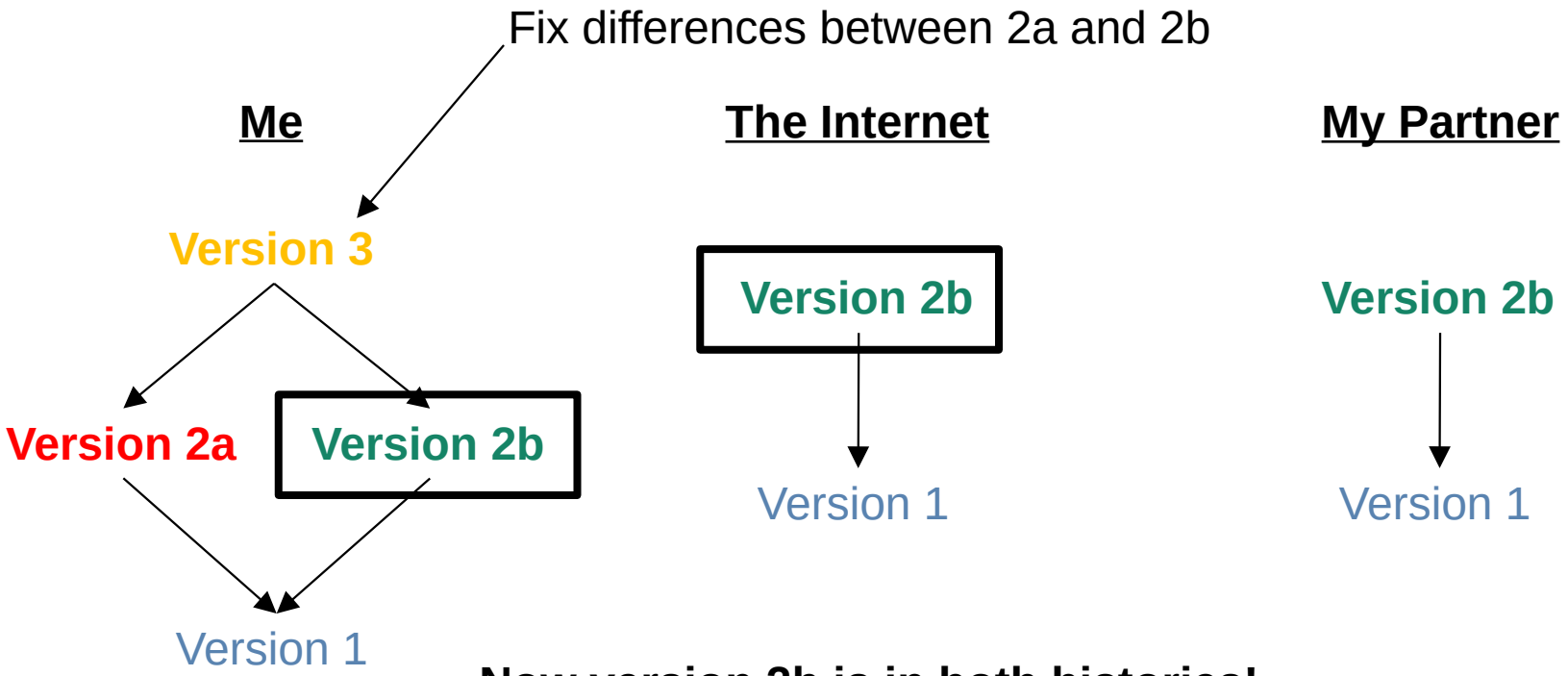
Version 1

My Partner

Version 2b

Version 1

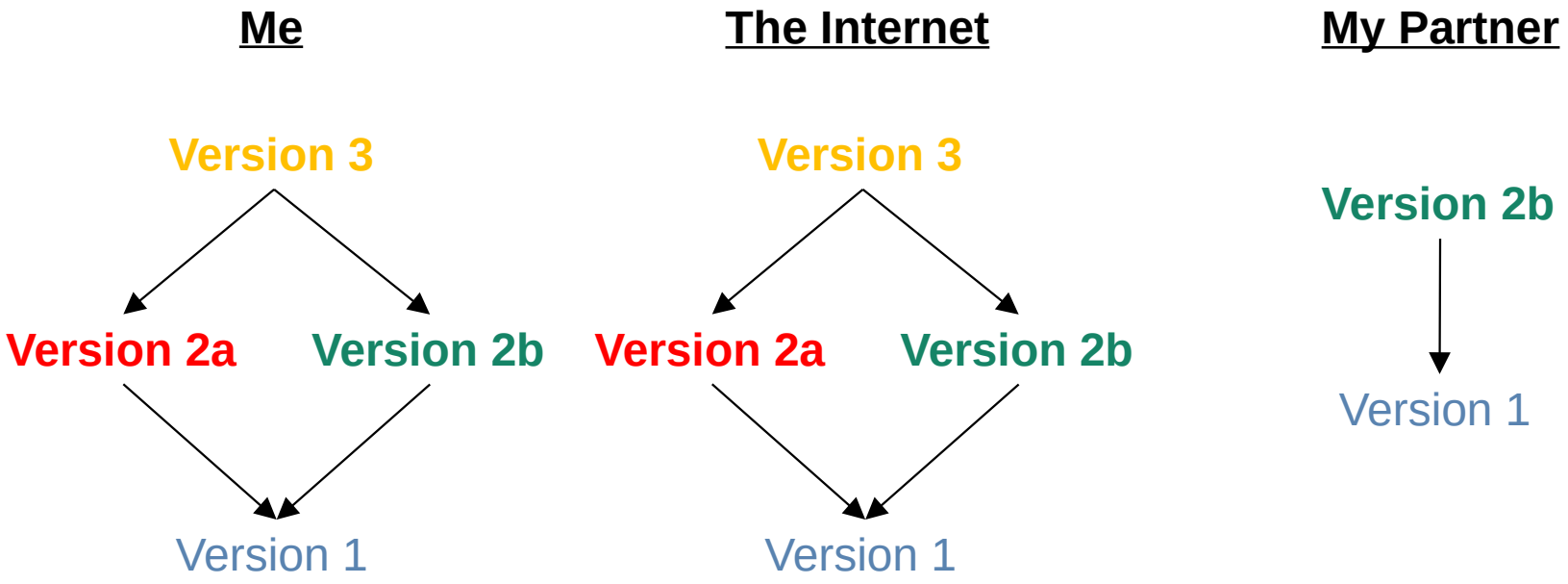
Multi-user cats



Now version 2b is in both histories!

How did we find it? BFS!

Multi-user cats



The surprise reveal...

Git is a big graph (a DAG)

- Vertices are:
  - Repository Versions (commits)
    - edges to previous commit(s)
    - edge to root directory
  - Directories (‘trees’)
    - edges to files/directories (labeled w/ name)
  - Files (blobs)
    - Leaf node, no out-edges

Things GIT does

- `git add`
  - Add new vertices/edges to the graph
  - Uses DFS to diff files if you add a directory
- `git commit`
  - Add a new version vertex to the graph
- `git push`
  - Compare the local head version with the remote's
    - If different, but remote head in history, diff (DFS) and upload
    - If different, but remote head not in history, require merge

Things GIT does

- `git pull`
  - Like push in reverse. Compare remote head with local
    - If same, no change
    - If different,
      - if local head in remote history, diff and download
      - if local head not in remote history, merge
        - Git tries to automatically merge files
          - If it fails, the user needs to fix things

Things GIT does

- `git diff <commit1> <commit2>`
  - Explicitly diff two versions (DFS)
- `git log`
  - Find all versions in the history (BFS)
- and lots more...

In Memoriam: Serenity [2012-2023]