Password Analyzer

All Pairs LCS Implementation

Caleb Donovick
David Morris

Problem Statement

- People use similar passwords for many services
- “Life Passwords” are very insecure

Longest Common Subsequence

Finds the longest common (not necessarily sequential) sequence amongst a pair of words.

- Ex:
  - X=ABCBDAB
  - Y=BDCABA
  - LCS(X,Y) = BDAB

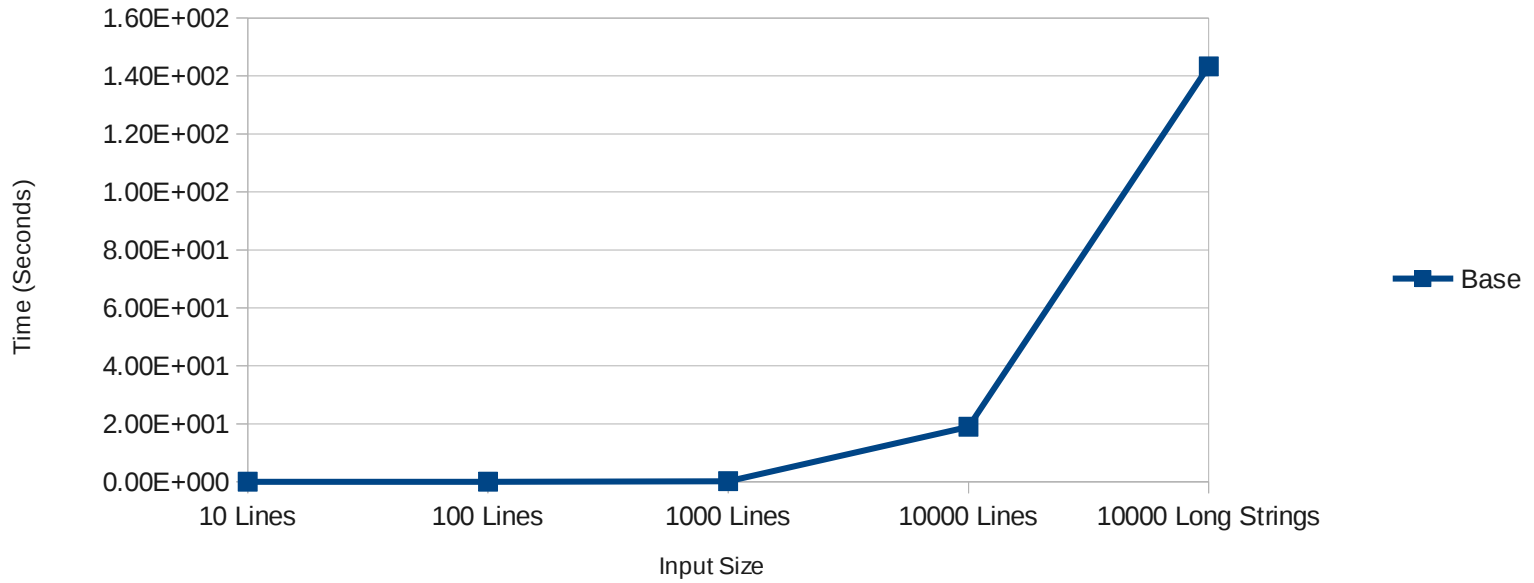
LCS Algorithm Breakdown

```
LCS(str a, str b):
 rows = length(a) + 1
 cols = length(b) + 1
  for i = 0 to rows:
   for j = 0 to cols:
    if i or j is 0:
     D[i, j] = 0
    else if  a[i-1] == b[j-1]
     D[i, j] = D[i-1, j-1] + 1
     P[i, j] = “U”
```

```
else:
 if D[i-1, j] > D[i, j-1]
  D[i, j] = D[i-1, j]
  P[i, j] = 'U'
 else:
  D[i, j] = D[i, j-1]
  P[i, j] = 'L'
```

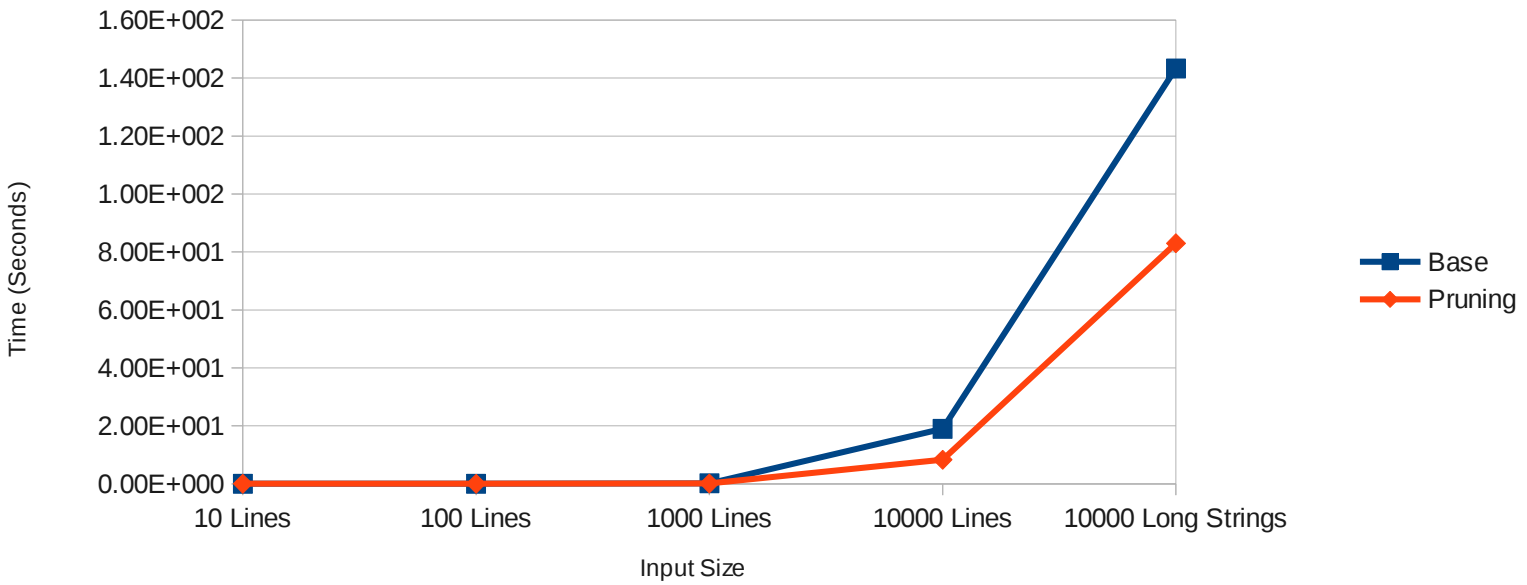
All Pairs LCS Base

- Brute Force check every option
- Complexity of O(n^2) * Complexity of LCS
- Can We Do Better?



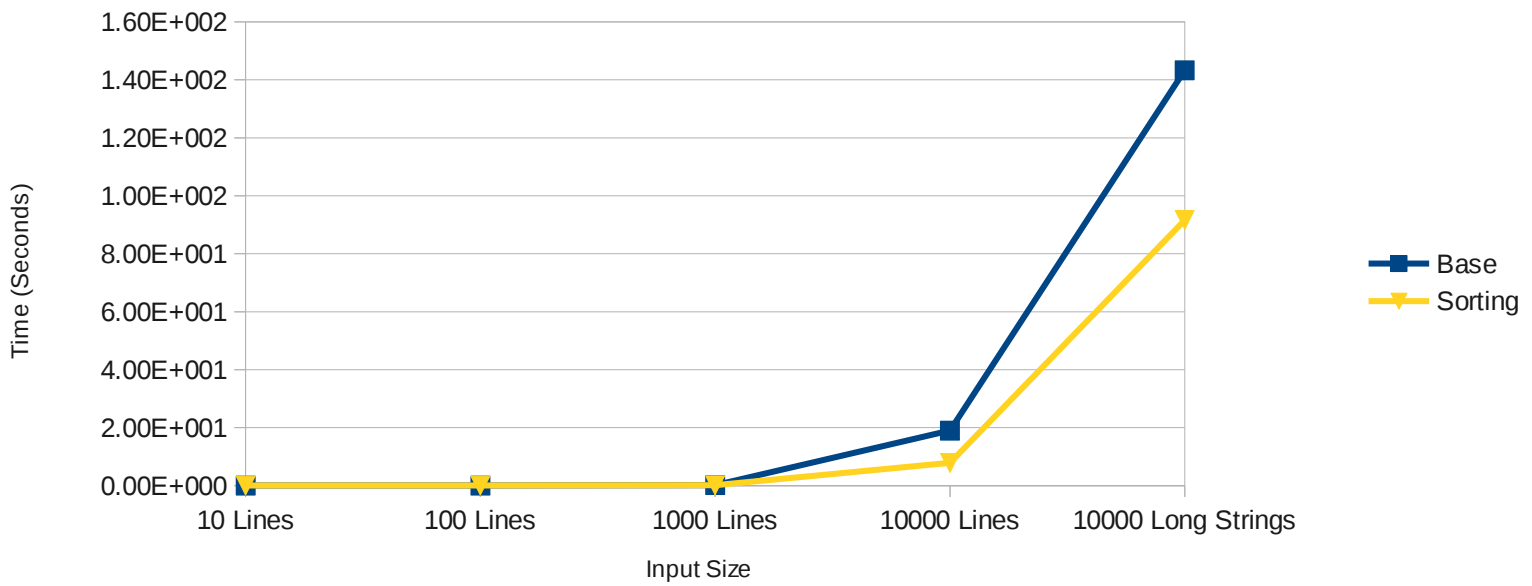
Pruning

- Choose not to explore strings that cannot offer a result
- Has option to abort LCS if not promising



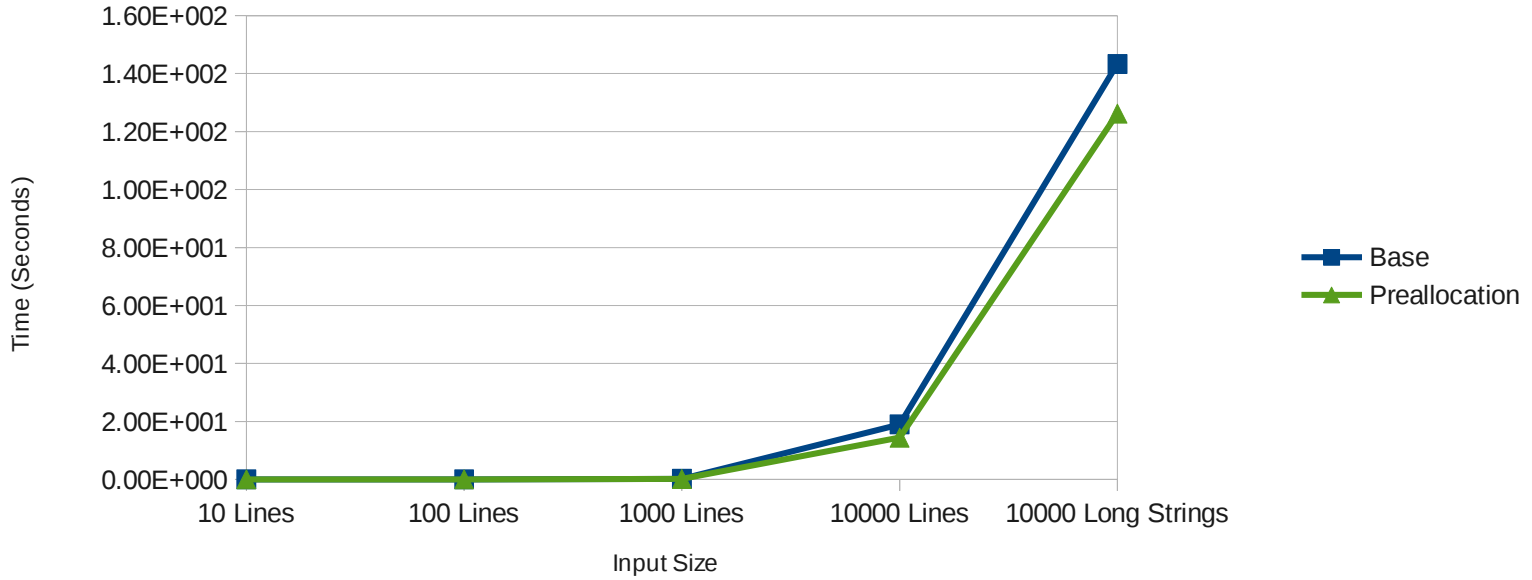
Sorting

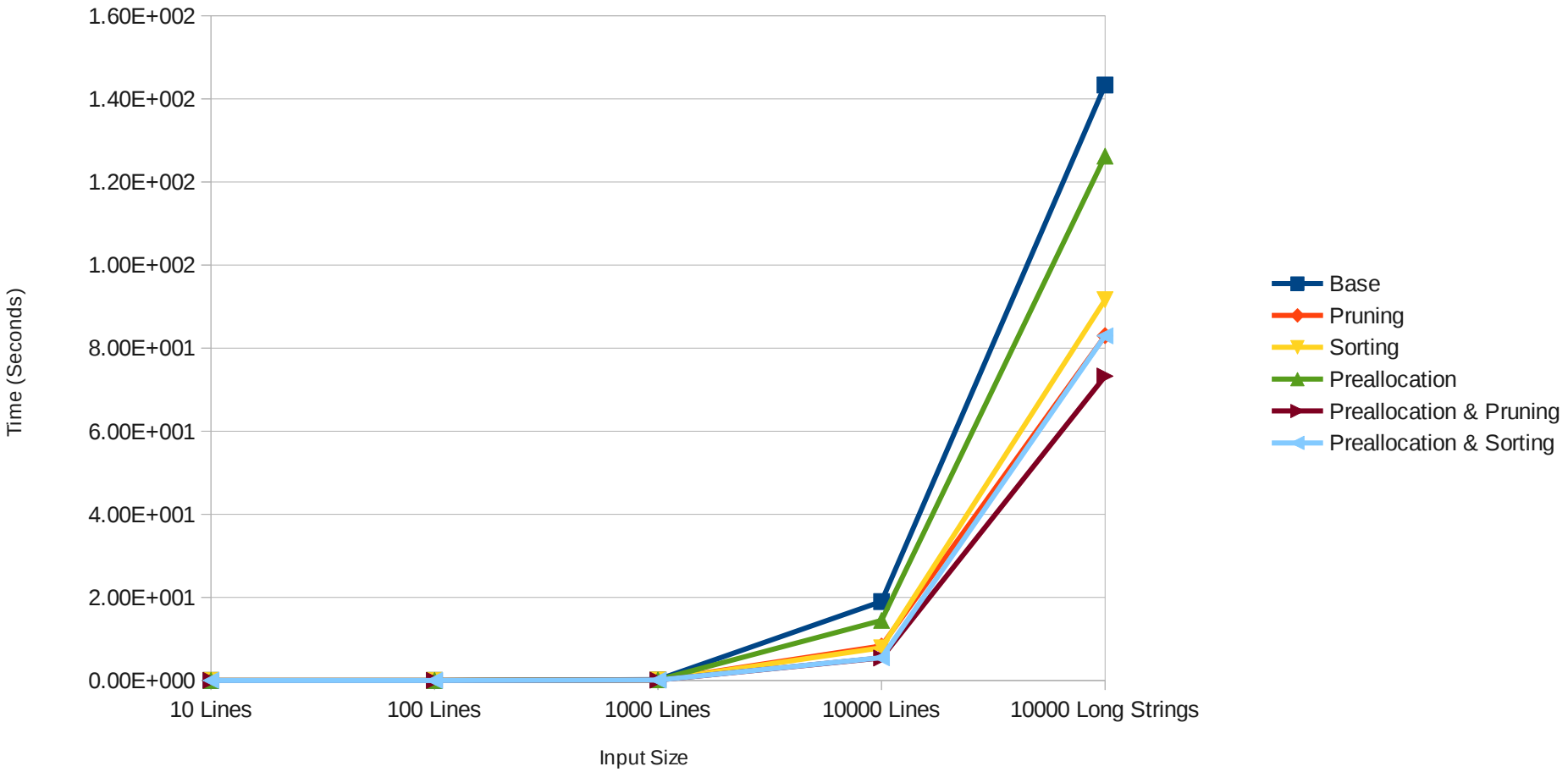
- Presort list of strings
- When a string that is shorter than current Worst LCS is found, exit AllPairsLCS



PreAllocation (And Modifications)

- Allocate space for tables before execution of algorithm
- Allows for batch memory allocation, and leading to lower runtimes and better cache efficiency




1.60E+002
1.40E+002
1.20E+002
1.00E+002
8.00E+001
6.00E+001
4.00E+001
2.00E+001
0.00E+000
Time (Seconds)
10 Lines
100 Lines
1000 Lines
10000 Lines
10000 Long Strings
Input Size
Base
Pruning
Sorting
Preallocation
Preallocation & Pruning
Preallocation & Sorting