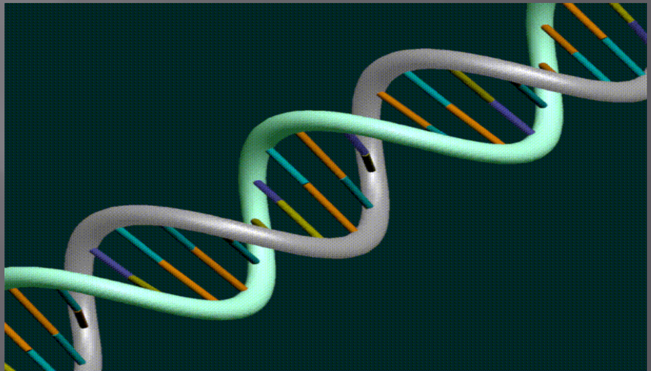
# DNA SCOUT

Brian Delgado
Nelson Ijih
Taibat Onaolapo Morakinyo

http://web.pdx.edu/~bdelgado/dnascout/

http://dnascout.codeplex.com (Subversion repository)



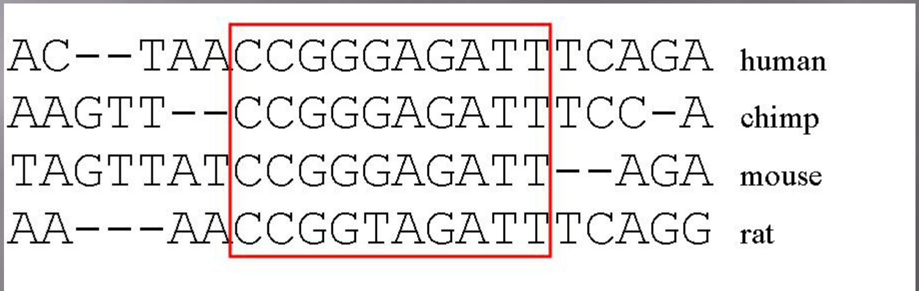
August 10, 2009

# Overview

- Project Overview
- Considerations
- Project Details
- Building the Tree
- Tree Node Re-use
- Sulfolobus Data Findings
- Partial Match
- Parallel Search
- Conclusion

# Project Overview

- Extension of CS 510 Multi-core programming project, under Dr. Karavanic, summer 2008.
- The goal of this project is to provide biological researchers with a tool to quickly search huge genome files for exact or partial pattern matches and report basic statistics regarding the matches.
- Potential uses:
  - Searching for common DNA patterns ("motifs") between related organisms. Motifs are subsequences that "have not mutated much over the course of evolution."
    - Motifs help the understanding of DNA since functional DNA evolves more slowly than non-functional DNA so motifs can help illustrate which parts of DNA are functional or non-functional.



```
AC--TAACCGGGAGATTTCAGA  human
AAGTT--CCGGGAGATTTCC-A  chimp
TAGTTATCCGGGAGATT--AGA  mouse
AA---AACCGGTAGATTTCAGG  rat
```

Aside: We share DNA with rats?

  - Searching for repeating DNA patterns in an organism and reporting how far apart in the sequence these matches are.
  - We are in contact with a BioInformatics researcher at OHSU who would like to see the output of this project.

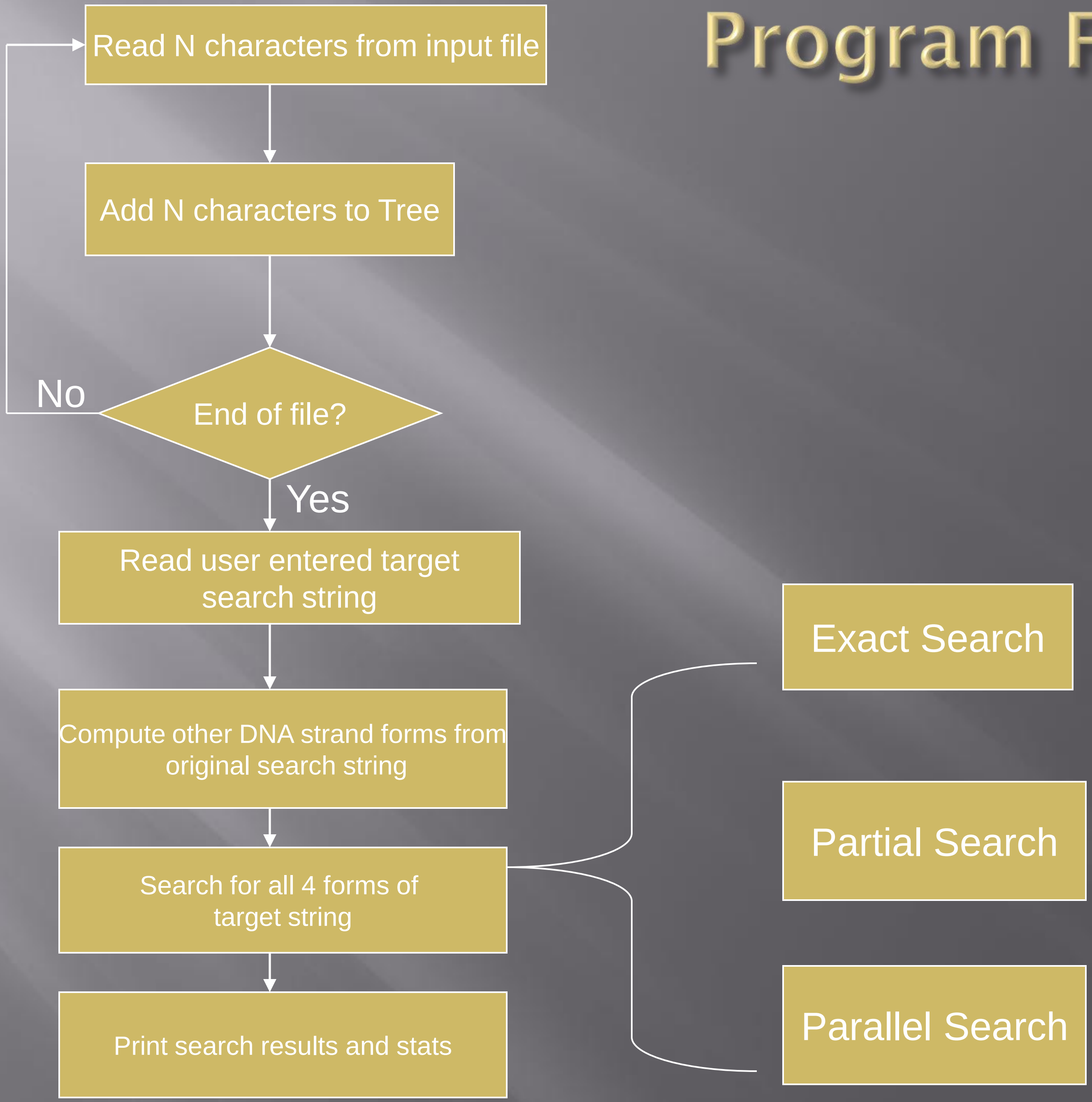
# Considerations

- **Getting the input data ready takes some manipulations**
  - DNA = interleaved helix of two strands
  - The NCBI database only has one strand of the DNA.
    - "CATATCTTAACGCGATTAATAAATACTCCGTATTTAAGAACTC…"
    - However, we can derive the second strand from the first strand using simple rules.
    - A <-> T
    - C <-> G
    - Also, need to flip both strands.
    - Total of four genome representations to search:
      - Strand 1 forward, e.g. "CAT"
      - Strand 1 reversed, e.g. "TAC"
      - Strand 2 forward, e.g. "GTA"
      - Strand 2 reversed, e.g. "ATG"
  - Memory Optimization: Instead of storing the four representations, we just manipulate the queries and store one representation.
- **Parallel code can help greatly.** With a quad core system, we could search the four representations concurrently.
- **Data structure efficiency** is extremely important. The inputs are so large that we need to be careful how we store the DNA input

# Project Details

- **DNA Scout supported features**:
  - 1. Create all sub-strings of length N from DNA file, store into memory search tree along with their location in the file.
  - 2. Exact match search (command line argument and text file input)
  - 3. Search for partial matches of sub-strings in the data-set.
  - 4. Parallel searches for multi-core systems.
  - 5. Statistics generated for each input file (distance between matching sub-strings, string frequency – i.e. how often does string X occur in the DNA file?)

# Program Flow



Read N characters from input file

Add N characters to Tree

End of file?

No

Yes

Read user entered target search string

Compute other DNA strand forms from original search string

Search for all 4 forms of target string

Print search results and stats

Exact Search

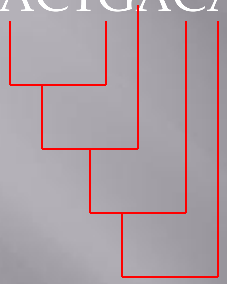
Partial Search

Parallel Search

# Building the Tree

Input: ACTGACATACTATT
Assume exact match length = 4

ACTGACATACTATT

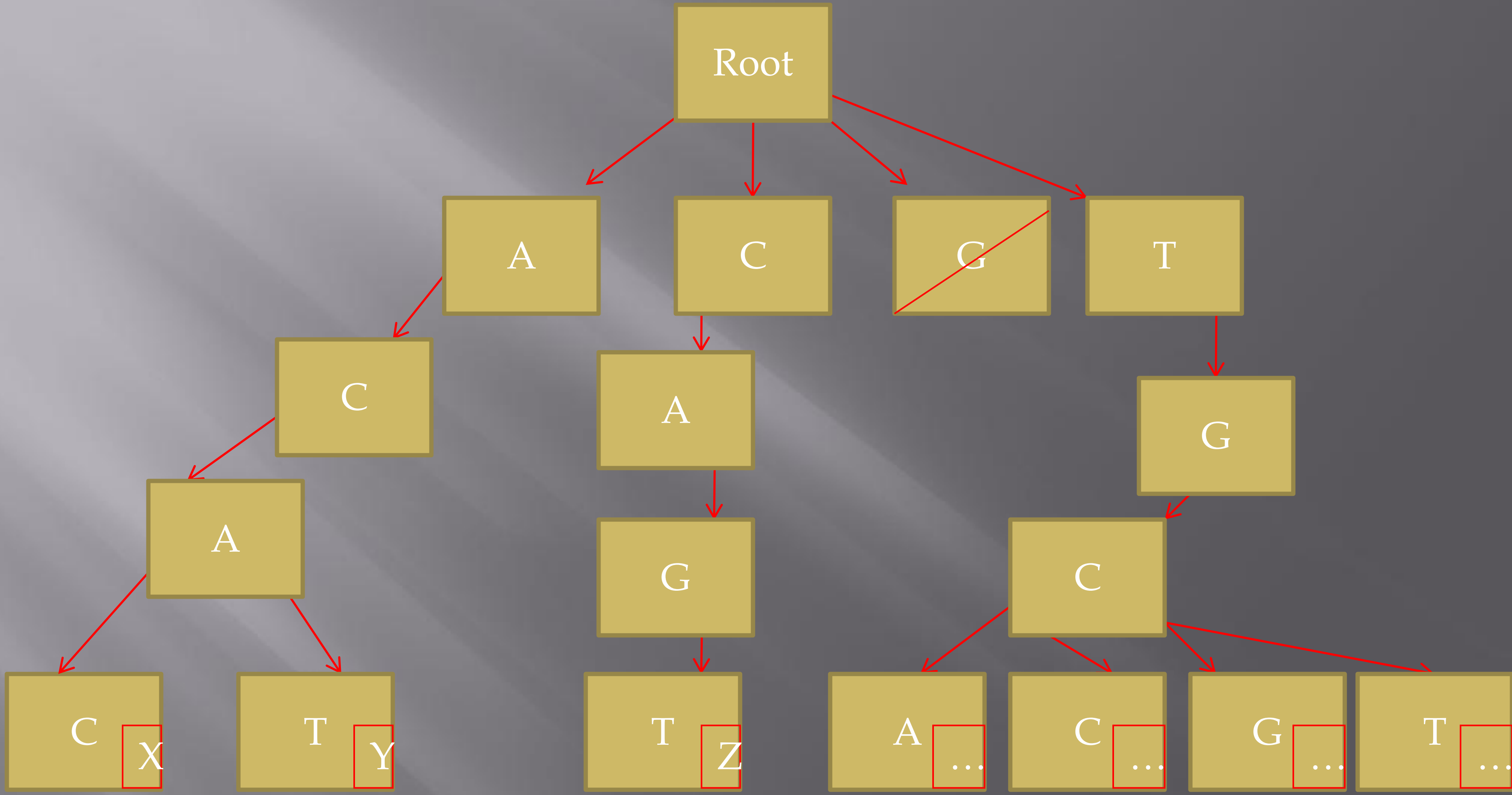


Start at first spot and take 4 characters (ACTG)
- Add to tree and note starting position (0) in tree
- Increment file pointer by 1 from starting location

Take next four characters (CTGA)
- Add to tree and note starting position (1) in tree
- Increment file pointer by 1 from starting location

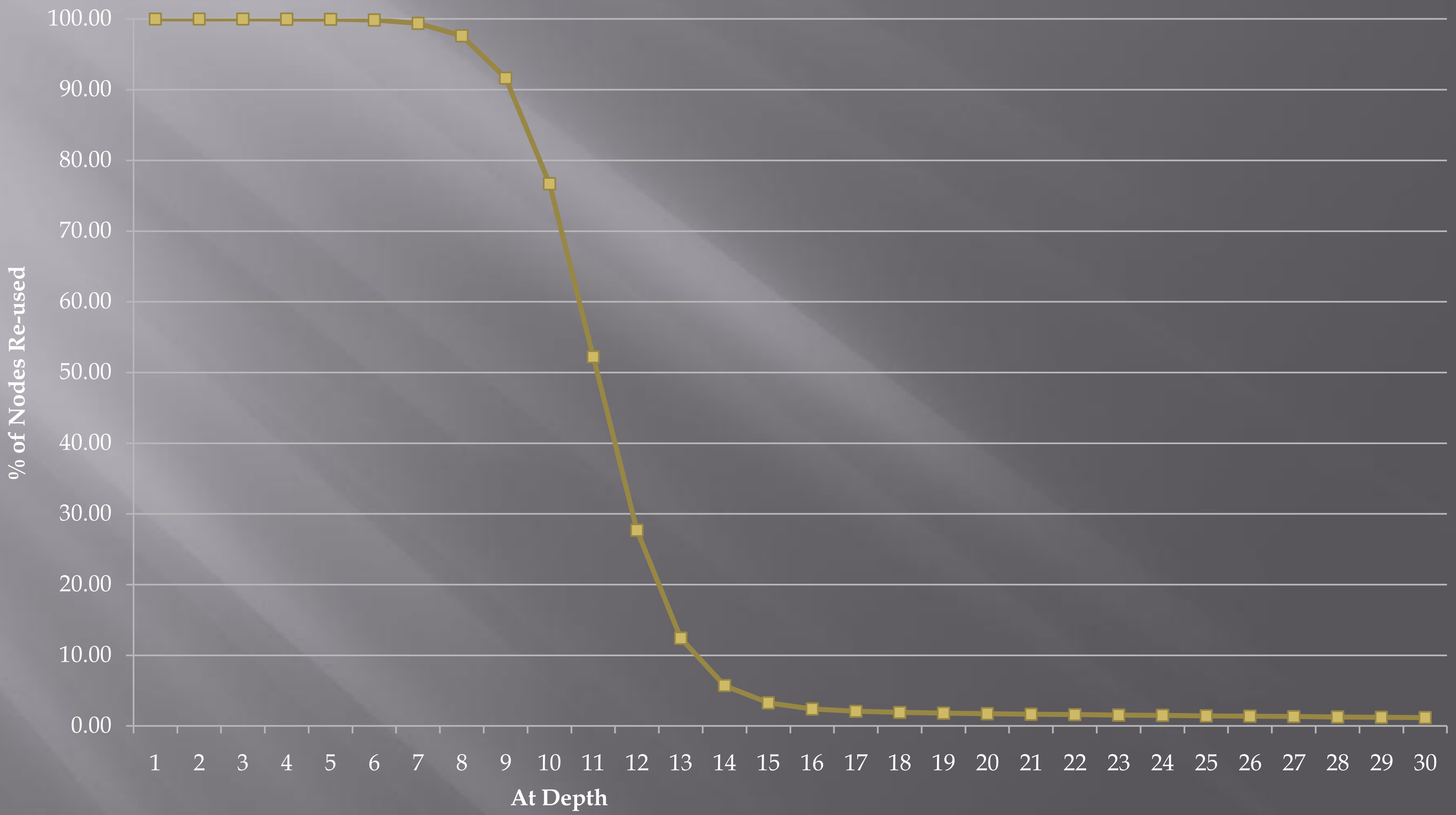
Repeat until file is read into the tree

# Tree Data Structure



This tree stores: {ACAC, ACAT, CAGT, TGCA, TGCC, TGCG, TGCT}
Null pointers are not shown, except for top-most G node.
The starting location of the string match is noted in the bottom-most node (e.g. X,Y,Z, etc)

# Instrumentation Results: Tree Node Re-use for Sulfolobus



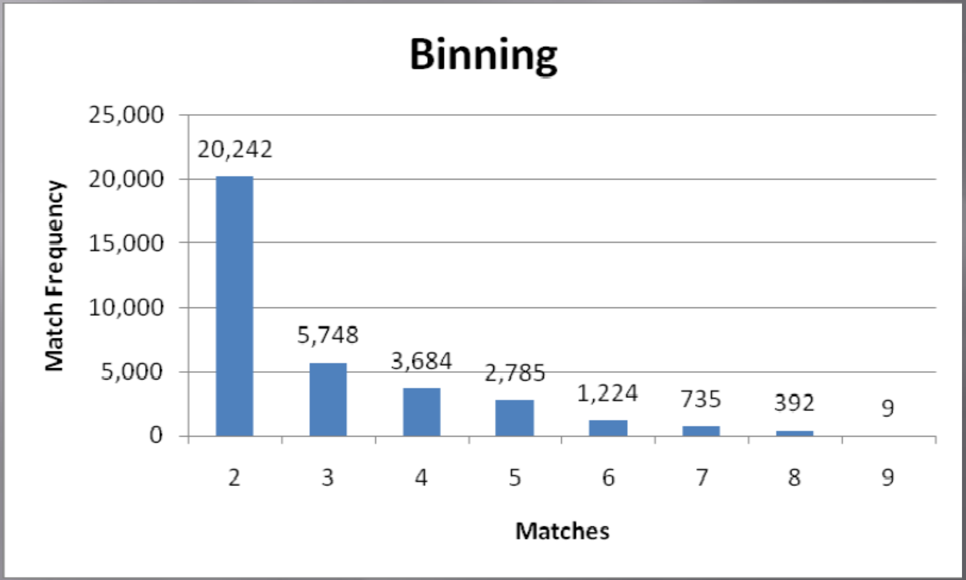
# Sulfolobus Data Findings

1. For Sulfolobus and a pattern match sizes of length 30, there are few redundant strings.

98.81% of all strings of length 30 are unique

1.19% of all strings of length 30 are redundant

2. Of the redundant strings, most of these matched twice in the genome although nine strings matched in nine different places.



3. Interesting, one long string (720 characters) matched in six places in the file.

# Partial Match Search

- Partial match searches are useful to researchers since DNA files may contain transcription errors or perhaps only differ in a few characters which would allow a partial match to be made.
- To search a partial DNA string from the original search string, e.g. from string “ACTATACGTAT”, can we find partial matches of the first N characters or between an arbitrary range.

- Needed to rebuild another search tree built from the tree built for original string

- Made the routine that builds tree modular and scalable to support partial search

- Added code to the build tree function to make sure it includes all characters from input file name to make sure partial searches results to hit if exist in tree

- Wrote a simple routine to get sub string from original search string

- Reported the position of the location in tree where sub search string is found.

# Parallel Search

- Useful in DNA Research
- Enhanced Searched Performance
- Example:
  - Given “ACTGCTGTAC” and want to find its permutation
    - Original string, reversed string, flipped string and flippedreversed will be handled by different threads
- Added master function that invoked search function
- Added code that split the search among threads
- Added a function that get the thread identification and report the position of the location in the tree where the permutation of the string is found

# Conclusion

- Future work:
  - More advanced partial match searches
    - Match strings which differ in N characters
  - Enhanced output, suitable for graphing in Excel or auto-generated graphs
  - Imprecise match searching, e.g. find matches which only differ by N characters

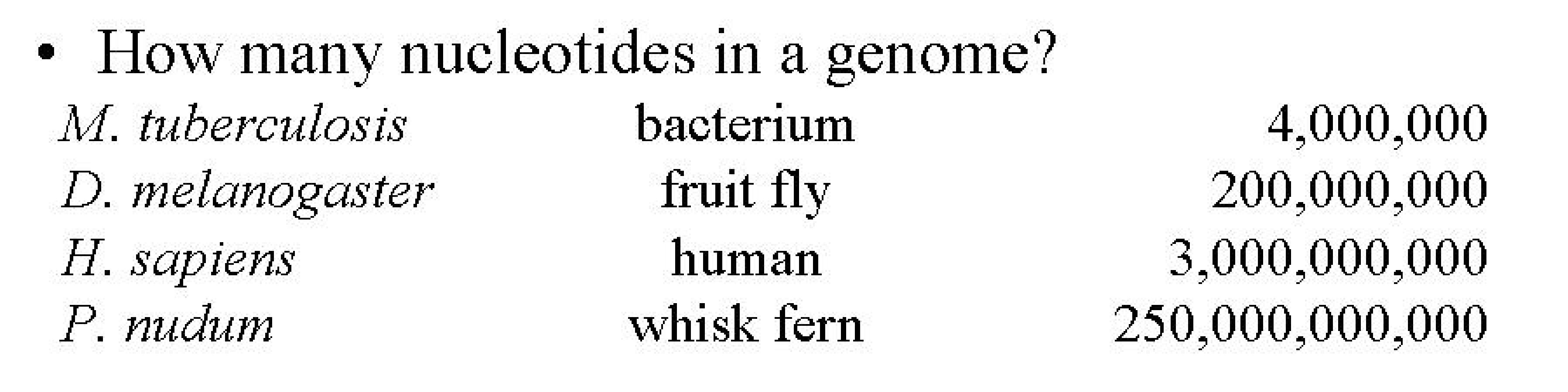
# Team Responsibilities and Acknowledgements

- **Team Responsibilities**
  - Brian Delgado: tree implementation, exact match search, file I/O, search permutations, stats and instrumentation
  - Nelson Ijih: partial match search, extensions to build tree
  - Taibat Onaolapo Morakinyo: parallel search, makefile

- **Acknowledgements**
- Dr. Karavanic, Greg Shauger, Dave Revell, John Ochsner for data structure discussions in Summer 08
- Dr. Bart Massey (Portland State Univ) for giving us the freedom to work on this project.
- DNA Image source: http://www.csb.yale.edu/userguides/graphics/ribbons/help/dna_rgb.gif

# Backup

- What are A,C,T,G?
  - **A**denine, **C**ytosine, **G**uanine, and **T**hymine
  - They are a nucleobase / DNA base. (parts of DNA/RNA that may be involved in pairing.)



- How many nucleotides in a genome?

| | | |
|---|---|---|
| *M. tuberculosis* | bacterium | 4,000,000 |
| *D. melanogaster* | fruit fly | 200,000,000 |
| *H. sapiens* | human | 3,000,000,000 |
| *P. nudum* | whisk fern | 250,000,000,000 |

# Tree Data Structures

```
// A node for a linked list of matches found in the DNA file for a given sub-string
struct matchList_str {
 int matchposition;
 matchList_str *next;
};

// Node in tree (i.e. the rectangle in the previous slide.)
struct node {
 struct node *anext;
 struct node *cnext;
 struct node *gnext;
 struct node *tnext;
 struct matchList_str *matchList;
};
```