Lucas Lee

CSE 13S · final notes

scan-build : compile time

- 2's complement = 1) Flip bits (0→1, 1→0) and add 1 to the result
  - ↳ 5: 0101 → 1010 + 1 = 1011 = additive inverse of 5

valgrind = run time

- Little Endian = |1 2|34|56|7 8| Big endian = |7 8|5 6|34|12|
- Pointers: Point to location of an object in memory (& = address-of operator)
  - ↳ don't add 2 pointers, multiply, or divide 2 pointers; Declaring an array allocates on stack ← dynamic allocation allocates on heap
  - ↳ Pointers to pointers for passing array of arrays → allows for multidimensional arrays (matrix)

fibonacci : $O(2^n)$ → recursion is slow

- Sorts = $O(n^2)$: Bubble, insertion, selection, (worst case) quick-sort
  - $O(n^{5/3})$ = Shell sort ; $O(n \log(n))$ = merge sort, heap sort, (average case) quick-sort
- insertion sort works best for already sorted arrays   build, fix heap ($\log n$), sort

Dynamic fib = $O(n)$ → best case fibonacci

- Doubly linked lists don't allow for random access, and sorting these linked lists checks each node 2 times $O(n^2)$
- Arrays allow random access in $O(1)$ time, linked lists allow sequential access
- Dynamic memory allocates at run-time on heap → stack space is limited → Slower to read/write to heap because of pointers
- Binary search = $O(\log(n))$, Only works on sorted array; Search algs use recursion often
- recursion is not always efficient, but is not inherently inefficient either
- Graphs: adding and checking for edge = $O(1)$; adjacency matrix = $O(n^2)$ space
  - ↳ adjacency list using linked lists : adding edge = $O(1)$, checking for edge = $O(n)$

makefile is just a DAG

- BFS = queue (level order), DFS = recursion or stacks
- Topological sort = DAG and makefiles : could be more than 1 topological order
- Makefile can include other makefiles

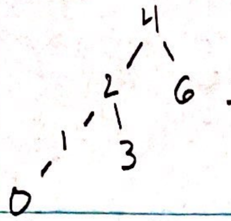
registers operate at nanosecond level

- Entropy - measures randomness - # of questions needed to guess symbols
  - ↳ ABCD > ABBC > AAAA
- Data compression algs: Huffman, LZ78 → uses tries and codes
- higher entropy messages → huffman > LZ78, low entropy - LZ78
- linked lists don't need to shift elements only change next / previous
  - ↳ not memory efficient → allocate memory for next nodes; no random access
  - ↳ doubly linked lists are even less memory efficient
  - ↳ lookups in linked lists are $O(n)$
  - ↳ to delete / insert, need to change where next / previous is pointing
  - ↳ you can make stacks and queues using linked lists, depending on where is head
  - ↳ doubly linked lists must have a head and tail pointing at eachother to start

context switching: save state by saving P.C., stack pointer, registers

multi threading →

- threads allow applications to do many things at once: faster than process - no address space
  - ↳ no 2 processes may be in critical regions, no assumptions about speed, no process outside of critical region can effect
    - ↳ process can't wait forever to enter critical region

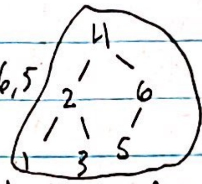


: imbalanced (height of left subtree is more than a difference of 1 from right)

binary search trees

file system can be represented as a tree

- Trees • DAG with nodes (- Preorder traversal = 4,2,1,3,6,5



Postorder : 1,3,2,5,6,4 - inorder : 1,2,3,4,5,6

↳ used in huffman to free all nodes → frees children before parents

level order traversal uses a queue - 4, 26, 1,3,5

link files using ln but cannot link directories

- balanced tree extrema search : $O(\log n)$ imbalanced : $O(n)$
- most files are opened using sequential access → lseek() for random access
- cannot write to directory - only rename, link (insert to), unlink (remove from)

↳ restricted operations on directories to prevent corruption of files

- Crypto - unbreakable code : one-time pad

↳ Public key cryptography for encrypting small amount of data

↳ Diffie-Hellman key exchange → exchange keys over insecure channels

↳ RSA - large primes (asgn5) - factoring large numbers is hard

$O(\sqrt{n})$ for factoring

↳ testing primes takes $O(\sqrt{n})$ divisions, and otherwise takes $O(n)$ time / space

forward referencing → by prototypes (headers)

↳ Probabalistic test (miller rabin) → if run 256 times, $1/2^{1000}$ chance to be wrong

↳ Polynomial deterministic time = $O(n^P)$, faster than $O(2^n)$ (P is large)

assembler takes .s files (assembly code) and converts it to binary (.o)

- Language translator : program converts high-level source code to lower level

↳ machine code interpreted in binary; → maps input to output

- Compilation in C : source code (.c) → Preprocessor → compiler → Assembly → object code → linker → executable

.o ↑ libraries ↓

header files, defines, etc. | lexical → regex, syntax → parsing, translation → convert to assembly, optimization → make code better, storage, code generator → make x86, assembly → makes .o files

- linker takes all .o file, resolves dependencies and connects them to create executables

cc calls -ld when it sees .o files

- Compilers translates programs all at once to assembly; Interpreters directly execute code one line at a time

code is faster for compiler, user experience faster for an interpreter

- gcc default on linux, clang = newest C compiler (default for macOS) standard compiler,
- Process is a program and the state of the CPU its running on → CPU runs 1 job at a time
- memory allocation: First fit, next fit, best fit, worst fit → best fit is the worst one
- Page table maps virtual addresses to physical addresses → Page number : Pointer to base address in physical memory

↳ each Process has its own page table; Physical memory mixes physical memory of multiple Process

POSIX threads = standard

- DFA - reconizes type 3 gramars (regular expressions) single-transition per letter - Same power as NFA
- NFA - many transitions per letter, PDA - context free languages + stack → NFA can be written as DFA

worst case DFA = 2^{NFA}

- Linear bound turing machine - linear bound tape + context-sensitive languages
- Turing machine - infinite tape, computes any computable function
- Open addressing + linked lists to counter hash collision → linear and quadratic probing + double hashing