CS 170 NOTES

LUKE YANG
NOTES FROM A COURSE BY AARON COTE

Abstract. These notes were taken during CS 171 (Discrete Methods in Computer Science) taught by Aaron Cote in Spring 2014 at University of Southern California. They were live-LaTeXed during lectures in `TeXShop` and compiled using `xelatex`. Each lecture gets its own section. The notes are not edited afterward, so there may be typos; please email corrections to `yifeiyan@usc.edu`.

1. Lecture 1, Monday, January 13

Course goals: Discrete math and problem solving skills with a wide range of topics.
Review: Sets, functions, sequences

Definition 1.1. A set is an unordered collection of objects.

Definition 1.2. Two sets are equal if and only if they have the same elements (aka objects, or members).

Venn graph
Universal set U contains all objects under consideration.

Definition 1.3. The intersection of sets S_1 and S_2, denoted by $S_1 \cap S_2$, is the set that contains those elements in both S_1 and S_2. $S_1 \cap S_2 = \{x | x \in S_1 \wedge x \in S_2\}$

Definition 1.4. The union of sets S_1 and S_2, denoted by $S_1 \cup S_2$, is the set that contains those elements that are either in S_1 or S_2, or both. $S_1 \cup S_2 = \{x | x \in S_1 \vee x \in S_2\}$

Definition 1.5. The complement of set S, denoted by $\overline{S}$, is the set that contains those elements that are in the universal set U but not in S. $\overline{S} = \{x | x \notin S\}$

Empty set $\emptyset = \overline{U}$

Definition 1.6. Two sets S_1 and S_2 are disjoint if $S_1 \cap S_2 = \emptyset$.

Generalized intersection

$$\bigcap_{i=1}^{n} S_i = S_1 + S_2 + \cdots + S_n$$

Generalized union

$$\bigcup_{i=1}^{n} S_i$$

Definition 1.7. Set A is a subset of set B, denoted by $A \subseteq B$, if and only if everything in A is in B.

Theorem 1.8. *Any set is a subset of itself. $\emptyset$ is a subset of any set.*

If two sets are subsets of each other, two sets are equal.

Definition 1.9. Set A is a strict subset of set B, denoted by $A \subset B$, if $A \subseteq B$ and $A \neq B$.

$A \subset B \Rightarrow A \subseteq B$, and its opposite is not true.

Example 1.10. 7 stamps: 2 red, 2 yellow, 3 green. A, B, C, are three perfect logicians. A removes blindfold and can't tell any conclusions about the colors about who's wearing what. B can't either.

From what A said, B and C can't wear red or yellow together. C wears green.

2. Lecture 2, Thursday, January 15

Example 2.1. 3 Truth machines are in stock. A machine corresponds true/false to red/green, but patterns for different machines can be different. 1 machine is broken (it answers arbitrarily) and 2 are working. Ask one single question (with a single yes/no answer) to one machine and determine which two are working.

Solution. Ask M1: is it the case that exactly one of these is true: Red means yes; 2nd machine is broken.

If Red, choose M2; if Green, choose M3.

Assume M1 works, and red means yes. Then it answers red if the 2nd machine works.

Assume M1 works, and green means yes. Then it answers green if the 2nd machine is broken.

Assume M1 works, both M2 and M3 work. Choosing either would be good.

Another method: meta-question: If I'd asked you, "Is machine 2 broken?", would you answer green?

Definition 2.2. The cardinality of a set A, denoted by $|A|$, is the number of distinct elements in A.

Example 2.3. $|\{\text{cake, pie, cake}\}| = 2$

Example 2.4. $|\emptyset| = |\{\}| = 0$

Example 2.5. $|\{\emptyset\}| = 1$

Example 2.6. $|\{\mathbf{Z}, \mathbf{N}\}| = 2$

Example 2.7. $|\mathbf{Z}| = \infty$

Definition 2.8. The power set of S, denoted by $P(S)$, is the set of all subsets of S.

Example 2.9. $S = \{\text{cake, pie}\}$, $P(S) = \{\emptyset, \{\text{cake}\}, \{\text{pie}\}, \{\text{cake, pie}\}\}$

Example 2.10. $P(\emptyset) = \{\emptyset\}$

Example 2.11. $P(\{\emptyset\}) = \{\emptyset, \{\emptyset\}\}$

Example 2.12. $P(P(\{a\})) = P(\{\emptyset, \{a\}\}) = \{\emptyset, \{\emptyset\}, \{\{a\}\}, \{\emptyset, \{a\}\}\}$

$P(A) = P(B) \to A = B$
$|A| = n \to |P(A)| = 2^n$

Definition 2.13. An ordered n-tuple is a collection of elements where order matters, i.e., an ordered set.

Example 2.14. $(a, b) \neq (b, a)$

Example 2.15. $(a, a, b) \neq (a, b)$

Definition 2.16. The Cartesian product of two sets A and B, denoted by $A \times B$, is an unordered set that consists all ordered pairs of (a, b) such that a is in A and b is in B.

$$A \times B = \{(a, b) | a \in A \wedge b \in B\}$$

Example 2.17. $\{1, 2\} \times \{1, 3, 4\} = \{(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4)\}$

Example 2.18. $\{(1, 1), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4)\} \times \{1, 2\} = \{((1, 1), 1), ((1, 3), 1), \cdots\}$

Definition 2.19. A function for A to B, denoted by $f : A \to B$, takes as input an element from set A and outputs an element from set B.

Definition 2.20. A function $f : A \to B$ is injective or one-to-one if every input maps to a distinct output, i.e., for an injective function f, $f(a) = f(b) \to a = b$

Example 2.21. $f(x) : \mathbf{R} \to \mathbf{Z}, f(x) = \lfloor x \rfloor$ is not injective.

Remark. Floor function $\lfloor x \rfloor$, ceiling function $\lceil x \rceil$.

Definition 2.22. A function $f : A \to B$ is surjective or onto if every element in B can be produced.

Example 2.23. $f(x) : \mathbf{R} \to \mathbf{Z}, f(x) = \lfloor x \rfloor$ is surjective.

Definition 2.24. A function f is bijective or one-to-one correspondence if it's both injective and surjective.

Example 2.25. $f(x) : \mathbf{Z} \to$ even integers, $f(x) = 2x$ is both injective and surjective, thus is bijective.

Definition 2.26. A sequence is a function from $\mathbf{N}$ to some set S.

Example 2.27. $f_0 = 0, f_1 = 1, f_2 = 2, f_3 = 3, f_4 = 5, f_5 = 8, \cdots$

Example 2.28. $1, 2, 3, 4, \cdots$

Example 2.29. $1, 4, 9, 16, \cdots$

Example 2.30. $f_n = f_{n-1} + f_{n-2},\ f_0 = 0,\ f_1 = 1$

Remark. Recurrence relations: recursive definitions of sequences.

Example 2.31. 3, 3, 3, 3, $\cdots$: $f_0 = 3,\ f_n = f_{n-1}$

Example 2.32. $f_n = 2n$: $f_n = 2 + f_{n-1}$

Example 2.33. $f_n = n^2$: $f_n = f_{n-1} + 2n - 1$

Example 2.34. $f_n = n + (-1)^n$: $f_n = f_{n-2} + 2$

Example 2.35. The polulation of world in 2010 is 6.9 billion, assume it grows at an annual rate of 11%. $f_n = 1.011 f_{n-1},\ f_0 = 6.9\text{billion}$

```
1 Function MergeSort(array A [1 : n])
2 if n == 1 then
3 |  return A
4 B = MergeSort (A [1 : ⌈n/2⌉])
5 C = MergeSort (B [⌈n/2⌉ + 1 : n])
6 return Merge (B, C)
```

3. Lecture 3, Wednesday, January 22

Running time analysis: analyzing sorting algorithms. Look into details of merge sort and selection sort.

Merge sort is better. Why? It has less total operations than selection sort. Count up the number of operations, but some operations take longer.

"if I double the size of input, how much does the runtime increase?"

$n \to$ double

$20n \to$ double

$10n + 37 \to$ double

large runtime $Cn+$ smaller terms $\to$ double

$n^2 \to 4\times$

$n^4 \to 16\times$

$n^5 + 10n^3 + 21 \to 32\times$

Polynomial times Cn^d+ smaller terms, increase by a constant factor 2^d

Selection fort, number of operations $\simeq n^2$

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \simeq n^2$$

Optimize this algorithm: Check if the list is sorted (n), it not, run Selection sort (n^2). We say n^2 because we analyze worst-case scenario.

In the average case, it'll be the same anyway (in this case it is true). $\frac{n^2+n}{2} \simeq n^2$. Must make guarantees. It's easier.

Analyzing growth rate of functions: look at worst case for large input sizes.

Definition 3.1. $f(n) = \mathrm{O}(g(n))$ "big-oh": $\forall n \geq n_0, f(n) \leq cg(n) \leftrightarrow \frac{f(n)}{g(n)} \leq c$, for some constants c, n_0.

Example 3.2. $10n^3 = \mathrm{O}(n^3)$

Example 3.3. $20n^2 + 13n + 5 = \mathrm{O}(n^2)$

Example 3.4. $10n^3 = \mathrm{O}(n^4) = \mathrm{O}(n^\infty)$

Constant factors are important, but they don't affect the growth rate.

O is imperfect knowledge of a function's running time. "Algorithm A is at least $\mathrm{O}(n^4)$" doesn't actually mean anything.

Definition 3.5. $f(n) = \Omega(g(n))$ "big-omega": $\forall n \geq n_0, f(n) \geq cg(n) \leftrightarrow \frac{f(n)}{g(n)} \geq c$, for some constants c, n_0.

Example 3.6. $10n^3 = \Omega(1) = \Omega(n^3)$

Example 3.7. $10n^2 + 10n + 2 = \Omega(n^2)$

Ω is imperfect knowledge as well. Neither is the best case nor worst case.
The optimized selection sort mentioned hereof is $O(n^2)$, and is $\Omega(n^2)$ as well.

Definition 3.8. $f(n) = \Theta(g(n))$ "big-theta" means $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Θ means perfect knowledge. Not every function has a Θ.
E-mail address: yifeiyan@usc.edu