Weekly Report(July.1,2018-July.8,2018)

Zhang Yuandi

Abstract

In the last week, I have finished the course *Divide and Conquer, Sorting and Searching, and Randomized Algorithms*, and learned week1 of *Machine Learning*. Besides, I have learned further about data structure and finished *Programming Foundations with JavaScript, HTML and CSS* to learn about HTML, CSS, and JavaScript.

1 Work done in these weeks

1.1 Algorithms

The courses discuss some classic algorithms with us.

1.1.1 Merge Sort

Merge sort is an efficient, general-purpose, comparison-based sorting algorithm. We can explain it as follows:

- recursively sort 1^{st} half of the input array
- recursively sort 2^{nd} half of the input array
- merge two sorted sublists into one

Merge Sort requires $\leq 6n \log_2 n + 6n$ operations to sort n numbers. It costs O($n \log n$).

1.1.2 Divide and Conquer

Divide and Conquer works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly.

Counting Inversions
Let's work on this problem:
Input: array A containing the numbers 1,2,3,..,n in some arbitrary order
Output: number of inversions = number of pairs (i, j) of array indices with $i < j$ and A[i] $>$ A[j]
We can solve it with this:

Sort-and-Count(array A, length n)
if n == 1, return 0
else
(B, X) = Sort-and-Count(1^{st} half of A, n/2)
(C, Y) = Sort-and-Count(2^{nd} half of A, n/2)
(D, Z) = CountSplitInv(A, n)
return X + Y + Z

It costs O(n$\log n$).

Matrix Multiplication
When we are doing matrix multiplication, divide and conquer always make sense.Let

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}, Y = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$$

Then,

$$XY = \begin{pmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{pmatrix}$$

With this algorithm, we can do matrix multiplication during a time which is better than cubic time.

Closest Pair
If we want to find out the pair of points that is closest, we can work like this:

ClosestPair(P_x, P_y)

1. Let Q = left half of P, R = right half of P. From Q_x, Q_y, R_x, R_y
2. (q_1, p_1) = ClosestPair(Q_x, Q_y)
3. (q_2, p_2) = ClosestPair(R_x, R_y)
4. (q_3, p_3) = ClosestSplitPair(P_x, p_y)
5. return best of (q_1, p_1), (q_2, p_2), (q_3, p_3)

It costs $n \log n$.

1.1.3 Quick Sort

Quick Sort is an efficient sorting algorithm, serving as a systematic method for placing the elements of an array in order.We can work like this:

Partition(A, l, r)

- p := A[l]
- i := l + 1
- for j = l+1 to r
- if A[j] $<$ p
  swap A[j] and A[i]
  i := i + 1
- swap A[l] and A[i-1]

It costs O(n).

1.1.4 Linear-Time Selection

Randomized Selection
Let's think about this problem:
Input: array A with n distinct numbers and a number
Output: ith order statistic
We can solve it with randomized selection:

Rselect(array A, length n, order statistic i)

1. If n == 1, return A[1]

2. Choose pivot p from A uniformly at random
3. Partition A around p let j = new index of p
4. If j == i, return p
5. If j > i, return Rselect(1^{st} part of A, j-1, i)
6. If j < i, return Rselect(2^{nd} part of A, n-j, i-j)

It costs O(n).

Deterministic Selection
To fix on the same problem as before, we can do this deterministic selection:

Dselect(array A, length n, order statistic i)

1. Break A into 5 groups, sort each group
2. C = the n/5 ”middle elements”
3. p = Dselect(C, n/5, n/10)
4. Partition A around p
5. If i == j, return p
6. If j < i, return Dselect(1^{st} part of A, j-1, i)
7. If j > i, return Dselect(2^{nd} part of A, n-j, i-j)

It costs O(n).

1.2 Machine Learning

The course in week1 helps me have a taste of machine learning, which seems mysterious and interesting to me. I have learned some essential knowledge of machine learning.

1.2.1 Definition

In Arthur Samuel’s opinion, machine learning is a field of study that gives computers the ability to learn without being explicitly programmed. In Tom Mitchell’s opinion, well-posed learning problem is that a computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E. Two main algorithms of machine learning is supervised learning and unsupervised learning.

1.2.2 Supervised Learning and Unsupervised Learning

Supervised learning is that the ”right answer is given”. It can be divided into regression learning(predict continuous valued output) and classification learning(predict discrete valued output). Unsupervised learning deals with ”unlabeled” data.

1.2.3 Cost Function

To measure how our algorithm works, we can define cost function to help us:

- Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$
- Parameters: θ_0, θ_1
- Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)} - y^{(i)})^2)$
- Goal: minimize $J(\theta_0, \theta_1)$

1.2.4 Gradient Descent

The basic idea is

- Start with θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

If there is just one single parameter, we can work like this
repeat until convergence

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) (for\ j = 0\ and\ j = 1)$$

Keep in mind that θ_0, θ_1 should be updated simultaneously.
And the linear regression case
repeat until convergence

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

Update both θ_0, θ_1 simultaneously as well.

1.3 Data Structure

I have learned three important trees. Those trees are very efficient when we confront with complex problems.

1.3.1 AVL Tree

An AVL tree is a self-balancing binary search tree. In an AVL tree, the heights of the two child subtrees of any node differ by at most one; if at any time they differ by more than one, rebalancing is done to restore this property.

1.3.2 Splay Tree

A splay tree is a self-adjusting binary search tree with the additional property that recently accessed elements are quick to access again. Splaying the tree for a certain element rearranges the tree so that the element is placed at the root of the tree.

1.3.3 RedBlack tree

A RedBlack tree is a kind of self-balancing binary search tree in computer science. Each node of the binary tree has an extra bit, and that bit is often interpreted as the color (red or black) of the node.

1.4 JavaScript, HTML and CSS

I have a lot of fun using JavaScript, HTML and CSS to make my web look nice. JavaScript is a high-level, interpreted programming language. HTML is the standard markup language for creating web pages and web applications. And CSS is a style sheet language used for describing the presentation of a document written in a markup language like HTML. The website CodePen is really cool. And I am trying to rewrite wordladder with JavaScript, although it seems quite a long way to go.

2 Plans for Next Week

1. Learn the course of week2, week3, week4 of **Machine Learning**.

2. Learn more about JavaScript and data structure.
3. Learn about detection networks.