به نام خدا

دانشگاه صنعتی امیرکبیر

دانشکده مهندسی برق

تمرین سری 6

Github

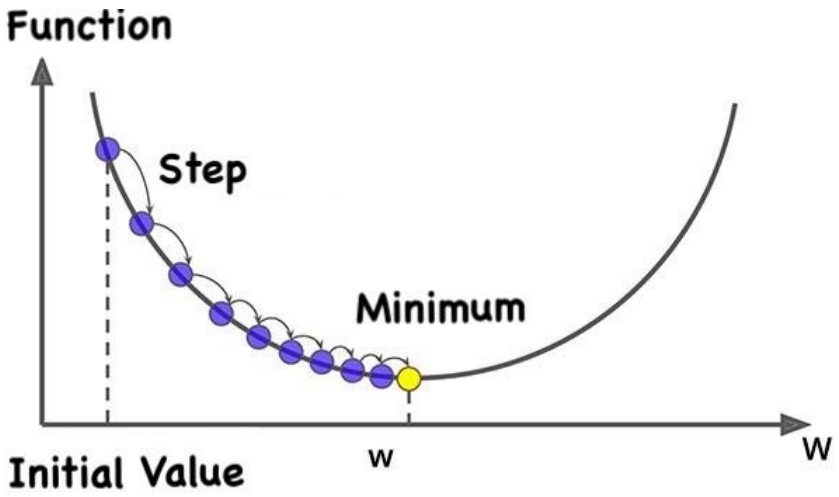
نام و نام خانوادگی	علیرضا عبدالله پوررستم
شماره دانشجویی	9823121
تاریخ ارسال گزارش	1401/03/14

فهرست گزارش سوالات (لطفاً پس از تکمیل گزارش، این فهرست را به‌روز کنید.)

سوال 1 – الگوریتم گرادیان کاهشی

این الگوریتم یک الگوریتم بهینه سازی برای حل عددی مسائل است.

این الگوریتم براساس نرخ تغییرات مشتق مرتبه اول عمل میکند.



برای این سوال در ابتدا یک تابع جدا به نام مشتق تعریف کردم :

که بر اساس مشتق حدی ، عمل گرادیان گرفتن را حساب میکرد.

$$\frac{f(x+h)-f(x-h)}{2h}=f'(x)$$

```
template <typename T, typename Func>
// define derviate function for calculate gradient_descent -> f'(x) = f(x+h)-f(x-h)/2*h and h goes to 0
double moshtagh(T point, Func func)
{
    double eps { 0.001 };
    double moshataghat { (func(point + eps) - func(point - eps)) / (2.0 * eps) };
    return moshataghat;
}
```

بررسی خود الگوریتم و کد آن :

```
template <typename T, typename Func>
double gradient_descent(T ivalue, double step, Func func = Func {})
{
    double learning_rate = { step };
    T iv { ivalue };
    double taghirat { learning_rate * moshtagh(iv, func) };
    while (std::abs(taghirat) >= 0.000000001) {
        taghirat = learning_rate * moshtagh(iv, func);
        iv = iv - taghirat;
    }
    return iv;
}
```

ابتدا نرخ یادگیری را با step مقدار دهی میکنیم ، سپس متغیری بنام تغییرات را به صورت ضرب نرخ یادگیری در مشتق تابع محاسبه میکنیم.

در آخر هم مقدار اولیه یا همان initial_value را عوض میکنیم.

چالش این سوال :

```
Func func = Func {})
```

در ابتدا کروشه را نگذاشته بودم که به صورت default عمل نکند اما در یکی از uni_test ها به دلیل تعریف function pointer کد دچار خطا میشد که یاد گرفتم اگر به صورت tamplate ای بخواهیم پاس بدهیم ، گذاشتن به صورت default مفید است.

سوال ۲ – regex روی دیتاست

در این سوال با regex آشنا شدم و بسیار برایم سخت بود. اما با کپی کردن کد ویدیو TA موفق به خواندن فایل و نمایش آن شدم.

مهم ترین بخش آن کار کردن با match ها بود. ما به تعداد گروه بندی هایمان ، match داریم.

```
std::vector<Patient> read_file(std::string filename)
{
    std::ifstream file(filename);
    std::stringstream buffer;
    buffer << file.rdbuf();
    std::string txt = buffer.str();

    std::regex pattern(R"((\w+)\ ?,(\w+)\ ?,(\d+)\,(\d+)\,(\d+)\,(\d+))");
    std::smatch match;

    std::vector<Patient> Patients {};

    while (std::regex_search(txt, match, pattern)) {
        std::string first_name { match[1] }, last_name { match[2] };
        std::string full_name { first_name + " " + last_name };
        // I use std::stoi for converting string to integer
        size_t age { static_cast<size_t>(std::stoi(match[3])) };
        size_t smokes { static_cast<size_t>(std::stoi(match[4])) };
        size_t area_q { static_cast<size_t>(std::stoi(match[5])) };
        size_t alkhol { static_cast<size_t>(std::stoi(match[6])) };
        Patients.push_back(Patient { full_name, age, smokes, area_q, alkhol });
        txt = match.suffix().str();
    }

    return Patients;
}
```

```
void sort(std::vector<Patient>& patients)
{
    std::sort(patients.begin(), patients.end(), comparison);
}
}
```

سوال 3 – regex رو فایل txt

این سوال regex پیچیده تری نسبت به سوال قبلی داشت، زیرا دارای match های زیادی بود.

و دلیل دیگر آن بودن زمان در فایل txt بود.

```
auto comparison { [](Flight first, Flight second) {
    return (first.connection_times + first.duration + 3 * first.price) > (second.connection_times + second.duration + 3 * second.
    price);
} };

auto gather_flights(std::string filename)
{
    std::priority_queue<Flight, std::vector<Flight>, decltype(comparison)> flights { comparison };
    std::ifstream file(filename);
    std::stringstream buffer;
    buffer << file.rdbuf();
    std::string txt = buffer.str();

    std::regex pattern(R"(\d+\- \w+\:(\w+)\ - \w+\:(\d+\h\d*\m*)\ - \w+\:(\d+)\ - \w+\:(\d+\h\d*\m*)\,?(\d*\h*\d*\m*)\,?
    (\d*\h*\d*\m*) - \w+\:(\d+))");
    std::smatch match;
    while (std::regex_search(txt, match, pattern)) {

        std::string flight_number { match[1] };
        size_t duration { transfer_to_time(match[2]) };
        size_t connections { static_cast<size_t>(std::stoi(match[3])) };
        size_t connection_times { transfer_to_time(match[4]) + transfer_to_time(match[5]) + transfer_to_time(match[6]) };
        size_t price { static_cast<size_t>(std::stoi(match[7])) };

        flights.push(Flight { flight_number, duration, connections, connection_times, price });
        txt = match.suffix().str();
    }
    return flights;
}
```

سوال 4 – functor

این سوال برای من زیباترین سوال تمرین بود.

این سوال در واقع Kalman filter بود. که دارای یک کلاس vector2D و دارای کلاسSensor بود.

در ابتدا این دو کلاس را با constructor هایشان initialize کردم و سپس به نوشتن functor برای پاس دادن به std::accumulate استفاده کردم.

در واقع یکی دیگر ازچیزهایی که این تمرین میخواست به ما یاد دهد آشنایی ما با std::accumalte بود.