Prof. Dr. Mine Elif KARSLIGİL BLM3021 Algoritma Analizi

**Son Teslim tarihi:** **11.12.2024**

Ödev #3

Oğuzhan Topaloğlu

Ç19052025 – Grup 1

*Bilgisayar Mühendisliği Bölümü,*

*Elektrik-Elektronik Fakültesi,*

*Yıldız Teknik Üniversitesi*

****

Istanbul, 2024

**Video Linki:** [**https://youtu.be/fLAY6pMXlgw**](https://youtu.be/fLAY6pMXlgw)

|  |
| --- |
| **İÇERİK** |
| 1. Problemin Çözümü |
| 2. Karşılaşılan Sorunlar |
| 3. Karmaşıklık Analizi |
| 4. Ekran Görüntüleri |
|  |
|  |
|  |
|  |
|  |

**1. Problemin Çözümü**

Problemi iki kısma ayırarak çözdüm: parsing ve hashing kısımları.

Parsing kısmında okunan dosya bir kez tur atılarak parse ediliyor ve değişken belirlendiğinde bir counter arttırılarak hashmap boyutu belirleniyor. Ardından ikinci bir tur daha atılarak oluşturulmuş hashmap’e bilgi girişi yapılıyor. Parsing yapılırken öncelikle “int main() {“ ifadesi bulununcaya dek boşluk gibi geçilecek karakterler geçilerek bir buffer’da string içinde okunan karakterler toplanıyor. Int main bulunduktan sonra da noktalı virgüllere göre satırlar ayrılıp iki kategoriye ayrılıyor. Birisi değişken tanımı yapılan “int x” ya da “char a,b,c” gibi satırlar iken öbürü ise herhangi bir aritmetik operasyonu içeren ve değişken tanımlamak yerine değişken erişimi yapan ve değişkenleri kullanan satırlar. Değişken tanımı yapılan satırlarda önceden deklere edilen bir değişken yeniden deklere edilmeye çalışıldıysa bir hata alınabilir, değişken kullanan satırlarda ise deklere edilmemiş bir değişken kullanıldıysa hata alınabilir.

Hashing kısmında ise double hashing algoritması ile sırayla tanımlamalar yapılıyor ve lookup ile insert fonksiyonları ile değişken tanımlama satırları ile değişken kullanma satırları çalıştırılıyor.

**2. Karşılaşılan Sorunlar**

Ödevde en çok zorlandığım kısım parsing oldu. Çoğu kez “int main() {“ ifadesi tam olarak bulunmadan değişik parsing işlemleri yapılarak null pointer exception’ı alıyordum. Ayrıca parsing’de bir sürü iç içe parsing olduğundan dolayı buffer’lar kafamı sıkça karıştırdı.

Örneğin tüm dosya içeriği bir buffer’da iken okunmuş kısım ayrı bir buffer’da ve bu okunmuş kısmın içinden değişken türü ve ismi çekiliyorsa bunlar da ayrı bir buffer’da…

Hashing kısmında pek sorun yaşamadım.

**3. Karmaşıklık Analizi**

Kodda bulunan fonksiyonların çoğu farklı karmaşıklığa sahiptir. Teker teker önemli fonksiyonları analiz etmek gerekirse,

1. Dosya içinde (okunan kodda) kaç karakter olduğunu bulan “long getFileLength(const char\*)” fonksiyonu O(1)dir. Bunun nedeni hesaplama için kullanılan fseek ve ftell fonksiyonlarının O(1) olmasından kaynaklanmaktadır.

2. Dosya içeriğini okuyan "char\* getFileContents(const char\*, long) fonksiyonu" O(n)'dir. Bunun nedeni, dosyanın tamamını okumak için kullanılan fread fonksiyonunun dosyadaki tüm karakterlere sırayla erişim yapmasıdır. Eğer dosya uzunluğu n ise, fread işlemi doğrudan dosyanın büyüklüğüne bağlıdır.

3. Hashmap boyutunu belirleyen "int determineHashmapSize(char\*, long, int)" fonksiyonu O(n)'dir. Bunun nedeni, kod içeriğinde tüm karakterlerin taranarak değişkenlerin belirlenmesi ve sayılmasıdır. Koddaki karakter sayısı n ise, her karakter sırasıyla kontrol edilir. Bununla birlikte, sonrasında asal sayıyı belirlemek için kullanılan isPrime fonksiyonu ek bir hesaplama gerektirir ancak n her zaman daha baskın olur.

4. Hashing fonksiyonu olan "int hash(Hashmap\*, char\*, int\*)" O(m)'dir. Bunun nedeni, verilen anahtarın uzunluğu m (string'in uzunluğu) ise, her bir karakterin sırayla işlenmesi gerektiğidir. Bu işlem sırasında hornerKey hesaplanarak bir indeks oluşturulur. Ancak bu, sadece anahtar uzunluğuna bağlı olduğu için sabit hash tablosu boyutuyla değişmez.

5. Hashmap'e değişken ekleyen "void hashmapInsert(Hashmap\*, char\*, char\*)" fonksiyonu worst case'de O(n) best case'de O(1) olarak çalışır. Bunun nedeni, double hashing kullanılarak indeks çakışmalarının etkili bir şekilde çözülmesidir. Ancak, en kötü durumda yani çok fazla çakışma varsa karmaşıklık O(n) olabilir.

**4. Ekran Görüntüleri**



















