1. ALGORİTMA KAVRAMI VE ALGORİTMALARIN İFADE EDİLME YÖNTEMLERİ

Giriş

Algoritma kavramı için, bilgisayarların bir problemi çözebilmesi için gerekli tüm adımları açıklayan bir tarif olarak düşünülebilir. Daha açık ifade edilmesi gerekirse, algoritmalar bilgisayara nelerin yapılması gerektiğini söyleyen talimatlar dizisidir. Hepimiz yemek kitaplarından veya internet üzerinden yemek tariflerini incelemişizdir. Bu tariflerde yemeği yapabilmemiz için gerekli malzeme ve yemeğinin nasıl yapılacağını adım adım açıklayan bir liste vardır. Algoritma da böyledir. Bir programlama algoritması, bir şeyin nasıl yapılacağını açıklar. Bu açıklamalar, bilgisayarınızın anladığı dile dönüştürüldüğünde, bilgisayarlar bunu her defasında açıklamalara uygun ve tam olarak çalıştırır.

Bugün algoritma konusu sadece bilgisayar bilimi ile uğraşanları değil, her alandan kişileri, kuruluşları da ilgilendiren bir konu olduğunu söylemek abartı olmayacaktır. Bundan birkaç kuşak önce algoritma denince insanların kafasında bir şey canlanmıyordu. Fakat bugün hangi işi yaparsak, hangi meslekten olursak olalım algoritmaları her gün her an kullanıyoruz. Algoritmalar her yerde. Bilgisayarlarımızda, Cep telefonumuzda, her gün kullandığımız asansörde, bindiğimiz otomobilde, uçakta, evde kullandığımız eşyaların birçoğunda.

Uzun zamandan beri teknolojisinin arkasında önemli algoritmalar yatan arama motorlarını kullanıyoruz ve bu şekilde istediğimiz bilgiye kolayca ulaşabiliyoruz. Alışverişlerimizin birçoğunu elektronik ticaret sitelerinden güvenli bir şekilde yapıyoruz. Çünkü bu siteler bizim için sır olan kredi kartı ve benzeri bilgilerimizi şifreleme algoritmaları ile İnternet üzerinden güvenli bir şekilde, üçüncü kişilerden koruyarak taşıyor. Bilgisayarlarımız, hata düzeltme algoritmalarını bize fark ettirmeden, her an kullanarak hayatımızı önemli ölçüde kolaylaştırıyor. Hayatımızda sıkıştırma algoritmaları olmasaydı verilerimizi internet üzerinden bir noktadan başka bir noktaya hızlı bir şekilde, daha az maliyetle nasıl gönderecektik?

Hayatımızda, yukarıda saydığımız algoritmalar gibi, en az onlar kadar etkili, yaygın kullanılan çok sayıda algoritma var ve bilim insanları her gün gece gündüz çalışarak bunlara yenilerini ekliyor. Bilgisayar bilimciler birbirlerinin çalışmalarını temel alarak, yeni problemlere çözüm olması için yeni algoritmalar geliştiriyor. Bu süreç hiçbir zaman olmadığı kadar baş döndürücü bir hızla devam ediyor. Algoritmalar hayatımızı, yaşama biçimimizi sarsıcı bir şekilde değiştirmeye devam ediyor.

1.1. Yaygın Olarak Kullanılan Algoritma Tanımlarından Bazıları

- Algoritma, belirli bir problemi çözmek için gerekli özgün adımlar serisidir (Dr. James F. Wirth).

- Matematikte ve Bilgisayar biliminde bir işi yapmak için tanımlanan, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumda sonlanan, sonlu elemanlar kümesidir(Wikipedia).

- Algoritma, bir problemi çözmek için gerekli yöntemi tanımlar (Goldschlager/Lister).

- Algoritma, belirli bir veri kümesi üzerinde kullanılacak olan aritmetiksel işlemlerin sırası ve türünü belirleyen kurallar kümesidir.

- Algoritma bilgisayara neleri yapması gerektiğini söyleyen talimatlar dizisidir.

- Algoritmalar, bir problemi sonlu zamanda çözebilmek için geliştirilmiş, açık, yürütülebilir, sıralı, basit ve gerektikçe tekrarlanan adımlardan oluşan yöntemlerdir.

- Belli bir problemi çözmek veya belirli bir amaca ulaşmak için tasarlanan yol(Wikipedia)

1.2. Algoritmaların Genel Özellikleri

a. Kesin Olmalıdır: Talimatlar bir bilgisayar tarafından yerine getirebilecek kadar kesin ve belirsizlikten uzak olmalıdır. Algoritmanın her adımı anlaşılır, şekilde ifade edilmiş olmalıdır.

b. Etkin Olmalıdır: Gereksiz tekrarlardan uzak olmalıdır. Birçok farklı problemin çözümünde kullanılabilecek genel özelliklere sahip olmalıdır. Örneğin, öğrencilerine yüzlük sistemde not veren bir üniversitenin, öğrenci işleri programının algoritması tasarlanırken yüzlük sistemin yanında dörtlük, beşlik vb. diğer sistemler de de kullanılacak şekilde bir algoritma tasarımı yapılmalıdır.

c. Sonluluk: Algoritma belirli sayıda adımdan oluşmalı, sonsuz döngüye(loop) girmemelidir.

d. Giriş bilgisine karşılık bir çıkış bilgisi sağlamalıdır: Algoritma, bir başlangıç durumundan başladığında, açıkça belirlenmiş bir son durumda sonlanan ve bu noktada bir giriş verisine karşılık bir çıkış verisi üreten özelliğe sahiptir.

e. Başarım ve Performans: Algoritmanın bilgisayarın belleğinde depolamaması gerek bilgi, bilgisayarın belleğinden fazla olmamalıdır. Gereksiz tekrarlar içermemelidir. Algoritmanın çalışma zamanının makul sürelerden fazla olmamalıdır.

1.3. Algoritma Dokümanlarında Olması Gereken Özellikler

a. Algoritma adı: Algoritmaya anlamlı bir isim verilmelidir.

b. Yaptığı iş: Algoritmanın Ne iş yapıldığı, kullanılan sabitler, değişkenler, sınıflar, nesneler hakkında yeterli bilgi bulunmalıdır.

c. İşlem adımları: Algoritmada işlem adımlarına numara verilmeli veya girintiler kullanarak anlaşılabilir olması sağlanmalıdır.

d. Açıklama: Algoritma metinlerinde yeterli açıklayıcı bilgi bulundurulmalıdır.

1.4. Algoritmaların İfade Edilme Şekilleri

Bir programlama algoritmasının, bilgisayar kodu olmadığını bilmek önemlidir. Algoritmalar basit bir şekilde, Satır Algoritması(Doğal dil) ile veya sözde kodlarla(Pseudocode) veya akış şemalarından(Flow Chart) biriyle ifade edilebilir.

Algoritmalar ifade edilirken, adımları numaralandırmak genellikle iyi bir fikirdir. Ancak, bazen algoritmalar ifade edilirken adımların numaralandırılma tekniği kullanılmaz. Bununun yerine algoritmanın okunabilirliğini arttırmak için, bazı satırlarının daha içeriden bazı satırlarının da daha dışarıdan yazılması yaklaşımı kullanılır.

1.4.1. Algoritmaların Satır Algoritmaları (Doğal Dil) İle İfade Edilmesi

Algoritmaların doğal dil(günlük yazı/konuşma dili) ile ifade edilme şeklidir. İfadenin doğal dil ile yapılmasından dolayı yazılması ve bu şekilde yazılan algoritmaların okunması, anlaşılması kolaydır. Satır algoritmalarda satır numaralarının belirtilmesi gerekmektedir.

Problem 1.1 Klavyeden kısa ve uzun kenarları girilen dikdörtgenin çevresini ve alanını hesaplayan algoritmanın satır algoritmasını yazınız?

Satır Algoritması

1. BAŞLA
2. YAZ "Uzun Kenar "
3. OKU ukenar
4. YAZ "Kısa Kenar "
5. OKU kkenar
6. Cevre=2*(ukenar + kkenar)
7. Alan=ukenar*kkenar
8. YAZ Cevre
9. YAZ Alan
10. SON

1.4.2. Algoritmaların Sözde Kod(Pseudocode) İle İfade Edilmesi

Sözde kod, herhangi bir katı programlama dili sözdizimi veya temel teknoloji hususları gerektirmeyen, gayri resmi bir programlama açıklaması yoludur. Bir programın taslağını oluşturmak için kullanılır. Sözde kod bir programın akışını özetler. Sistem tasarımcılarının, programcıların bir yazılım projesinin gereksinimlerini anlamalarını ve kodu buna göre geliştirmelerini sağlamak için sözde kod yazılır. Sözde kod program ile algoritma veya akış şeması arasında köprü görevi görür.

Sözde kod gerçek bir programlama dili değildir. Bu nedenle yürütülebilir bir programda derlenemez. Sözde kodlar, Algoritmalar belirli bir programlama diline dönüştürülmeden önce, kısa terimler veya basit İngilizce sözdizimleri kullanılarak hazırlanır. Bu, üst düzey akış hatalarını belirlemek ve program geliştiricilerin veri akışlarını anlamaları için yapılır. Kavramsal hatalar sözde kodlar hazırlanırken önceden düzeltildiği için bu, gerçek programlama sırasında kesinlikle zaman kazanmanıza yardımcı olur.

Sözde kodun hazırlamanın ana amacı, bir programın her satırının tam olarak ne yapması gerektiğini açıklamaktır. Sözde kod aşamasında hataları yakalamak veya yanlış program akışlarını tespit etmek, bu hataları daha sonra yakalamaktan daha az maliyetli olduğundan, programların kodlanması aşamasına geçmeden önce sözde kod hazırlamak faydalıdır. Sözde kod hatalardan arındırılıp kabul edildiğinde, bir programlama dilinin kelime ve sözdizimi kullanılarak program yazılmasına geçilir.

İp Ucu:

Sözde Kod;

1. *Yürütülebilir bir programda derlenemez*
2. *Programcıların sadece kod geliştirme sürecinin algoritma kısmına odaklanmasını sağlar.*
3. *Her tür programcı tarafından anlaşılır.*

Problem 1.2 Klavyeden kısa ve uzun kenarları girilen dikdörtgenin çevresini ve alanını hesaplayan algoritmanın sözde kodunu yazınız?

Sözde Kod

```
1. Begin
2.    Print "Uzun Kenar ";
3.    Read ukenar;
4.    Print "Kısa Kenar ";
5.    Read kkenar;
6.    Cevre =2*(ukenar + kkenar);
7.    Alan=ukenar*kkenar;
8.     Print Cevre;
9.     Print Alan;
10. End
```

1.4.3. Algoritmaların Akış Şemaları İle İfade Edilmesi

Akış şemaları, algoritmaları görsel olarak temsil etmek için ideal diyagramlardır. Farklı alanlardaki bir süreci veya diyagramı analiz etmek, tasarlamak, belgelemek veya yönetmek için kullanılırlar. Diğer diyagram türlerine benzer şekilde, neler olduğunu görselleştirmeye yardımcı olurlar. Bu, bir süreci anlamaya ve içindeki kusurları ve darboğazları bulmaya yardımcı olur.

Program akış şeması, bir binanın mimari planına benzer. Bir tasarımcı, bir bina inşa etmeye başlamadan önce mimarı plan çizer. Aynı şekilde, bir programcı akış şemasına dayalı bir bilgisayar programı yazmadan önce bir akış şeması çizer. Tıpkı bir plan çizmek gibi, akış şeması, Amerikan Ulusal Standart Enstitüsü, tarafından verilen standart akış şeması sembollerini içeren tanımlanmış kurallara göre çizilir. MS Word veya hatta MS Visio'da akış şemalarını kolayca yapabilirsiniz (nasıl kullanılacağı hakkında daha fazla bilgi edinebilirsiniz)

Bir akış şeması, belirli bir sorunun çözümünü elde etmek için gerçekleştirilecek işlem sırasını gösteren diyagramlara sahiptir. Programcılar ve analistler arasında iletişimi sağlar. Bir akış şeması çizildikten sonra, programı herhangi bir üst dilde yazmak nispeten daha kolay hale gelir. Başka bir deyişle, karmaşık bir programın iyi belgelenmesi için akış şemaları zorunludur. Algoritmalar temsil edilirken iki farklı akış şeması çizilebilir: Yüksek Seviye Akış Şeması, Ayrıntılı Akış Şeması.

Yüksek Seviye Akış Şeması

Bu akış şeması, bir algoritmada önemli adımları göstermektedir. Ayrıca her bir adımın ara çıktılarını ve ilgili alt adımları verir. Bundan başka, sürecin temel bir resmini sunar ve süreç içinde meydana gelen değişiklikleri tanımlar.

Ayrıntılı Akış Şeması

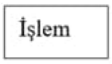
Bu akış şeması, süreçteki tüm adımlar ve faaliyetler dâhil sürecin ayrıntılı bir resmini verir. Sürecin adımlarını ayrıntılı bir şekilde incelemek ve sorunları veya verimsizlik alanlarını saptamak için çok yararlıdır.

Akış Şeması Sembolleri

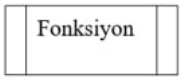
Başlangıç ve Bitiş Sembolleri: Genellikle daireler, oval şekiller veya yuvarlatılmış dikdörtgenlerle gösterilir. Normalde, başlangıç sembolünün içine "BAŞLAT" sözcüğü yazılır. Son sembolün içinde "SON" sözcüğü yazılır.

Oklar: Kontrol akışını gösterirler. Bir ok bir simgeden gelip başka bir simgeyle sona erdiğinde, program kontrolünün okun işaret ettiği simgeye geçeceği anlamına gelir.

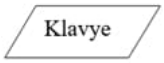
Dikdörtgenler: Bir eylemi veya işlemi temsil ederler.



Dış Fonksiyon: Genellikle çift dikey kenarlı dikdörtgenlerle gösterilirler.



Girdi: Genellikle paralelkenarlar ile gösterilir.



Koşul veya Karar: Elmas olarak tasvir edilmiştir. Bir ok girişi, iki ok çıkışı vardır. Bir ok "Evet" veya "Doğru" anlamına gelir. Diğer ok “Hayır” veya “Yanlış” a karşılık gelir.

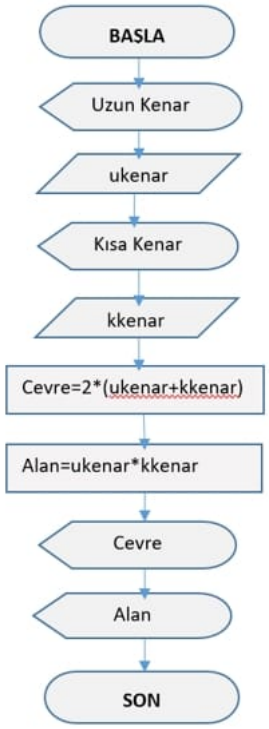
Bağlayıcılar: Bunlar bir dairenin içindeki tanımlayıcı bir etiketle gösterilir. Etiketli konektörler ok yerine çok sayfalı diyagramlarda kullanılır.

Görüntüleme: Kâğıt ve ekrana yapılacak çıktıların sembolleri

Döngü: Döngülerin gösterimi için kullanılan sembol

Manyetik Disk: Veritabanı sembolü

Problem 1.2 Klavyeden kısa ve uzun kenarları girilen dikdörtgenin çevresini ve alanını hesaplayan algoritmanın akış şemasını çiziniz?



Uygulamalar

Ekrana Merhaba C++ Yazdıran Program

Ekrana Merhaba C++ yazılmasını sağlayan algoritmanın satır algoritmasını, sözde kodunu, akış şemasını ve C++ kodunu yazınız?

a. Satır Algoritması

1. BAŞLA

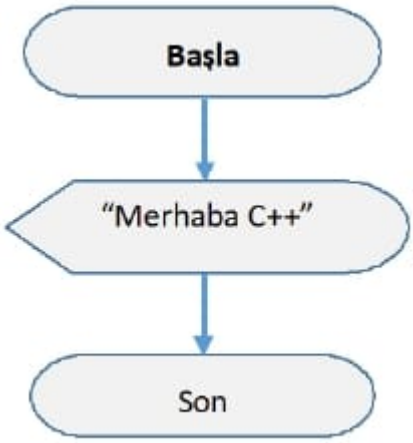
2. YAZ “Merhaba C++”

3. SON

b. Sözde Kod

```
1. Begin
2.    print  “Merhaba C++”;
3. End
```

c. Akış Şeması



d. C++ Kodu

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Merhaba C++ \n";
    return 0;
}
```

Tam Sayı Tipi Değişken Tanımlamak, Değişkene Değer Atamak ve Toplama Yapmak

Tamsayı türünde iki değişkenin tanımlanan ve bu değişkenlere değer atanan algoritmanın satır algoritmasını, sözde kodunu, akış şemasını ve C++ kodunu yazınız?

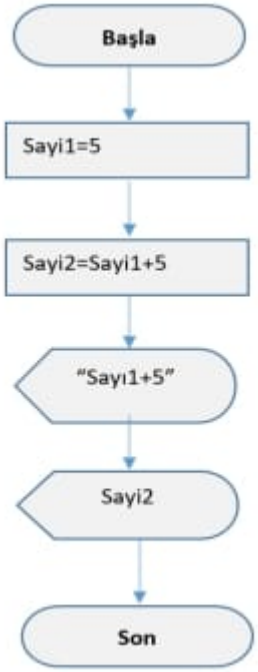
a. Satır Algoritma

1. BAŞLA
2. Sayi1=10;
3. Sayi2=sayi+5
4. YAZ "Sayi1+5="
5. YAZ sayi2
6. SON

b. Sözde Kod

```
1. Begin
2.   Sayi1=5;
3.   Sayi2=Sayi1+5;
4.   Print "Sayi1+5=";
5.   Print sayi2;
6. End
```

c. Akış şeması



d. C++ Kodu

```
#include <iostream>
using namespace std;
int main()
{
    int sayi1; // sayi1 değişkeni tanımlanıyor
    int sayi2; // sayi2 değişkeni tanımlanıyor
    sayi1 = 10; // sayi1 değişkenine 10 değeri atanıyor
    sayi2 = sayi1 + 5; // sayi2 değişkenine değer atanıyor

    cout << "sayi1 + 5 = "; // metin yazdırılıyor
    cout << sayi2 << endl; // sayi2'nin değeri yazdırılıyor
    /* endl alt satıra geçmek için kullanılır
    "\n" yazmak ile aynıdır  */
    return 0;
}
```

Klavyeden Girilen İki Sayıyı Toplamak

Klavyeden girilen iki sayıyı toplayan ve sonucu ekrana yazdıran algoritmanın satır algoritmasını, sözde kodunu, akış şemasını ve C++ kodunu yazınız?

a. Satır Algoritması

1. BAŞLA

2. YAZ “Birinci Sayıyı Giriniz “

3. OKU birinci sayı

4. YAZ “İkinci Sayıyı Giriniz “

5. OKU ikinci sayı

6. Toplam=birinci sayı + ikinci Sayı

7. YAZ Toplam

8. SON

b. Sözde Kod

```
1. Begin
2.      Print “Birinci sayıyı Giriniz “;
3.     Read Sayı1;
4.      Print “İkinci sayıyı Giriniz “;
5.      Read sayı2;
6.     Toplam=sayi1+Sayi2;
7.     Print Toplam;
8. End
```

c. Akış Şeması

d. C++ Kodu

```
#include <iostream>
using namespace std;
int main( )
{
    int sayi1,sayi2,toplam;
    cout<<"Birinci Sayıyı Giriniz :";
    cin>>sayi1;
    cout<<"İkinci  Sayıyı Giriniz :";
    cin>>sayi2;
    toplam=sayi1+sayi2;
    cout<<toplam<<endl;
    return 0;   }
```

Klavyeden Girilen Fahrenheit Cinsinden Sıcaklığı Santigrad Cinsine Çeviren Program

Klavyeden girilen fahrenheit cinsinden sıcaklığı santigrat cinsine çeviren, sonucu ekrana yazdıran algoritmanın satır algoritmasını, sözde kodunu, akış şemasını ve C++ kodunu yazınız.

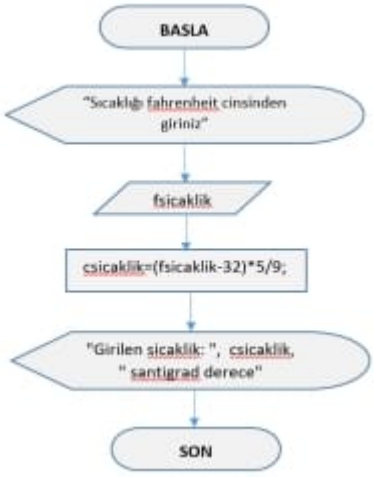
a. Satır Algoritma

1. BAŞLA
2. YAZ "Sicakligi fahrenheit cinsinden giriniz:”
3. OKU fsicaklik
4. csicaklik=(fsicaklik-32)*5/9
5. YAZ "Girilen sicaklik: ", csicaklik, " santigrad derece"
6. SON

b. Sözde Kod

1. Begin
2. Print “Sicakliği fahrenheit cinsinden giriniz”;
3. Read fsicaklik;
4. csicaklik=(fsicaklik-32)*5/9;
5. Print "Girilen sicaklik: ", csicaklik, " santigrad derece";
6. End

c. Akış Şeması



d. C++ Kodu

```
#include <iostream>
 using namespace std;
  int main()
  {
int fsicaklik;
// Ekrana mesaj yazılıyor
cout << "Sicakligi fahrenheit cinsinden giriniz: ";
// Klavyeden girilen değer okunuyor
cin >> fsicaklik;
// Girilen değer Santigrad cinsine çevriliyor
int csicaklik = (fsicaklik - 32) * 5 / 9;
// Sonuç ekrana yazılıyor
cout << "Girilen sicaklik: " << csicaklik < " santigrad derece" << endl;
return 0;
  }
```

1.5. Algoritma Geliştirme Adımları

Bir algoritmanın (bir planın) geliştirilmesi, bir problemin çözümünde önemli bir adımdır. Bir algoritmaya sahip olduktan sonra, onu bazı programlama dillerinde bir bilgisayar programına çevirebiliriz. Algoritma geliştirme sürecimiz beş ana adımdan oluşmaktadır. Bu adımlar; problemi tanımlamak, analiz, üst düzey algoritma geliştirmek, Algoritmayı hassaslaştırmak, algoritmayı test etmek.

1. Problemi Tanımlamak: Bu adım göründüğünden çok daha zordur. Etkin, başarımı ve performansı yüksek bir algoritma geliştirmek için, problemin analizi yapan gurup tarafından tam olarak anlaşılması gerekir. Aksi durumda beklenen algoritma geliştirilmesi ve buna bağlı olarak doğru çalışan bir bilgisayar programı yazılması imkânsızdır.

Algoritma geliştirmenin bu aşamasında müşteri, problemin tanımını oluşturmaktan sorumludur. Ancak bu genellikle sürecin en zayıf kısmıdır. Zayıflık, tanımının aşağıdaki hata türlerinden bir veya daha fazlasından olumsuz etkilenmesinden kaynaklanmaktadır: (1) açıklama, belirtilmemiş varsayımlara dayanır, (2) açıklama belirsizdir, (3) açıklama eksiktir veya (4) açıklamanın iç çelişkileri vardır. Bu eksiklik, sorunu tanımlayan müşterinin dikkatsizliğinden veya kullanılan dilin belirsizliğinden kaynaklanabilir.

Analiz grubunun sorumluluğunun önemli bir kısmı, problemin tanımlanası sırasında yapılan hataları tespit etmek ve hataları gidermek için müşteri ile çalışmaktır.

2. Analiz: Analiz adımının amacı, problemi çözmek için hem başlangıç hem de bitiş noktasını belirlemektir. Bu süreç, neyin verildiğini ve neyin kanıtlanması gerektiğini belirleyen bir matematikçinin yaşadığı sürece benzer. İyi bir problem tanımı, bu adımı gerçekleştirmeyi kolaylaştırır.

Başlangıç noktasını belirlerken, aşağıdaki sorulara cevap aranarak başlanabilir: Hangi veriler mevcut? Bu veriler nerede? Problemle ilgili hangi formüller var? Verilerle çalışmak için hangi kurallar var? Veri değerleri arasında hangi ilişkiler vardır?

Bitiş noktasını belirlerken, çözümün özelliklerini tanımlamamız gerekir. Başka bir deyişle, işimiz bittiğini nasıl bileceğiz? Aşağıdaki soruları sormak genellikle bitiş noktasını belirlemeye yardımcı olur: Ne gibi yeni gerçeklerimiz olacak? Hangi öğeler değişecek? Bu öğelerde ne gibi değişiklikler yapılacak? Artık hangi şeyler olmayacak?

3. Üst Düzey Algoritma Geliştirmek: Algoritma, bir sorunu çözmek için bir plandır ve bu planlar birkaç ayrıntı düzeyinde yapılabilir. Algoritmanın geliştirilmesine çözümün büyük bölümünü içeren yüksek seviyeli bir algoritma hazırlamakla başlamak genellikle daha iyidir, ancak ayrıntılar daha sonra ele alınmalıdır.

4. Algoritmayı Hassaslaştırmak: Üst düzey bir algoritma, bir sorunu çözmek için izlenmesi gereken önemli adımları gösterir. Algoritma geliştirmenin bu dördüncü adımında algoritmaya ayrıntı eklememiz gerekiyor. Ancak, ne kadar ayrıntı eklemeliyiz? Ne yazık ki, bu sorunun cevabı duruma bağlıdır.

Bu adımda amacımız, bilgisayar programı yazarken kullanacağımız algoritma geliştirmek olduğundan, programlayıcının yeteneğine uyacak detay seviyesini ayarlamamız gerekir. Şüphe duyduğumuzda veya öğrenirken, çok az ayrıntıya sahip olmaktan, çok fazla ayrıntıya sahip olmak her zaman daha iyidir.

Basit problemlerin çözümü için geliştirilen algoritmalarda, tek bir adımda üst düzey algoritmadan ayrıntılı bir algoritmaya geçilebilir. Ancak, bu her zaman mümkün olmayabilir. Daha büyük, daha karmaşık problemler için, bu işlemi birkaç kez tekrarlamak, önce orta seviye algoritmalar geliştirmek gerekir. Bu tekrarların her defasında, önceki algoritmaya daha fazla ayrıntı eklemeliyiz, daha fazla ayrıntıya ihtiyaç kalmadığı noktada durmalıyız.

5. Algoritmanın Sınanması: Son adım algoritmayı sınamaktır. Ne arıyoruz? İlk olarak, geliştirilen algoritmanın orijinal problemi çözüp çözemeyeceğini belirlemeliyiz. Bunun için algoritma üzerinde adım adım çalışmamız gerekir. Algoritmanın probleme bir çözüm sağladığı konusunda tatmin olduktan sonra başka şeyler aramaya başlamalıyız. Aşağıdaki sorular, bir algoritmayı test etmek için sorulabilecek sorulara örnek olabilir.

Geliştirilen algoritma çok çok özel bir problemi mi çözüyor, yoksa daha genel bir problemi mi çözüyor? Çok çok özel bir problemi çözüyorsa, genelleştirilmeli mi? Örneğin, 3 metrelik yarıçapa sahip bir dairenin çevresini hesaplayan bir algoritma (formül 2 *π * 3) çok özel bir problemi çözer, ancak herhangi bir dairenin çevresini (formül 2 * π * r) hesaplayan bir algoritma daha genel bir problemi çözer.

Bu algoritma basitleştirilebilir mi? Bir dikdörtgenin çevresini hesaplamak için bir formül: uzunluk + genişlik + uzunluk + genişlik, daha basit bir formül şu şekilde olabilir: 2.0 * (uzunluk + genişlik)

Bu çözüm başka bir sorunun çözümüne benziyor mu? Nasıl benzerlikler var? Nasıl farklılar var? Örneğin, aşağıdaki iki formülü göz önünde bulunduralım: Dikdörtgen alan = uzunluk * genişlik Üçgen alanı = 0.5 * taban * yükseklik. *Benzerlikler:* Her biri bir alan hesaplar. Her biri iki ölçümü çarpar. *Farklılıklar:* Farklı ölçümler kullanılır. Üçgen formülü 0.5 içerir. *Hipotez:* Belki de her alan formülü iki ölçümün çarpılması ile hesaplanabilir.

Algoritmanın sınanma işlemi, bu sorular çoğaltılarak ve cevapları tespit edilerek tamamlanır ve bu adım geçildikten sonra bilgisayar programcılarına programlarını geliştirirken kullanacakları kusursuz bir algoritma dokümanı sunulur.

Bölüm Özeti

Bu bölümde, günlük hayatımızda kullandığımız algoritmaların hayatımızı nasıl kolaylaştırdığını, algoritmaların tanımını, başarımı ve performansı yüksek, etkin bir algoritmanın genel özelliklerini ve de algoritma dokümanlarında bulunması gereken özellikleri öğrendik.

Algoritmaların ifade edilmesi için kullanılan üç ayrı yöntem(Satır algoritması, Sözde kod, Akış şeması) konusunda bilgi sahibi olduk ve örneklerle konuyu pekiştirdik.

Bir algoritma geliştirme döngüsündeki farklı adımları öğrenerek, basit algoritmalar hazırlayacak beceri kazandık.

Tufan Göbekçin(Çeviri), Mater Algoritma, Paloma Yayınevi, 2017.

Kadir Çamoğlu, Algoritma, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2011.

H. Burak Tungut, Algoritma ve Programlama Mantığı, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2019.

Muhammed Mastar, Süha Eriş, C++, KODLAB Yayın Dağıtım Yazılım ve Eğitim Hizmetleri San. ve Tic. Ltd. Şti, 2012.

Şadi Evren Şeker(Çeviri Editörü), Algoritmalar, Nobel Yayınevi, 2018.