Veri Yapıları ve Algoritmalar Ödev 3

Öğrenci Adı: Fatih Alaybeyoğlu

Öğrenci Numarası: 23011009

Dersin Eğitmeni: M. Elif Karslıgil

Video Linki: https://youtu.be/8UEMl3pAVu0

1 – Problemin Çözümü:

Bu programda, bir şirkete ait personel yapısı ağaç (tree) veri yapısı kullanılarak modellenmektedir. Her çalışan bir TreeNode yapısı ile temsil edilir; bu yapının içinde çalışanın adı, yaşı, maaşı ve hiyerarşik bağlantılar için sameLevel (aynı seviyede olan kardeş çalışan) ve underLevel (bu çalışana bağlı olan ast çalışanlar) göstericileri bulunur. Böylece hiyerarşik bir yapı içinde gezinmek ve analiz yapmak mümkün olur.

Ağaç yapısının kurulması, dışarıdan sağlanan bir dosya aracılığıyla gerçekleştirilir. readFile fonksiyonu, bu dosyayı satır satır okuyarak her satırdaki çalışanın adını, yaşını, maaşını ve bağlı olduğu patronun adını alır. Eğer patron adı "NULL" olarak belirtilmişse, o çalışan kök düğüm olarak kabul edilir. Diğer çalışanlar, push fonksiyonu kullanılarak uygun patronun altına eklenir. Bu işlem için önce findByName fonksiyonu çalıştırılarak patron bulunur, ardından yeni çalışan düğümü oluşturularak patronun underLevel bağlantısına bağlanır. Eğer patronun halihazırda başka astları varsa, yeni çalışan bu listenin başına eklenir ve eski alt çalışanlar sameLevel üzerinden devam ettirilir. Böylece her çalışanın astları underLevel yoluyla bir bağlı liste (linked list) gibi sıralanır.

Veri analizi aşamasında, ilk olarak hiyerarşik yapının toplam derinliği (seviye sayısı) howManyLevels fonksiyonu ile hesaplanır. Bu fonksiyon, tüm ast seviyelere özyinelemeli (recursive) olarak giderek her seviyeyi sayar. Algoritmik olarak, her çalışanın underLevel bağlantısı üzerinden alt seviyeye inilmekte, sameLevel bağlantısı üzerinden ise aynı seviyedeki diğer çalışanlar gezilmektedir. Böylece ağaç derinlemesine ve yatayına doğru taranmış olur.

Her seviyedeki çalışan sayısını hesaplamak için howManyEmployeeAtEachLevel fonksiyonu kullanılır. Bu fonksiyon, yine özyinelemeli çalışarak her düğümde ilgili seviye için sayaç değerini artırır. level parametresiyle derinlik tutulur ve employeeForLevels dizisinin ilgili indisi güncellenir. Örneğin level = 0 olduğunda kök seviyedeki çalışan sayısı artırılır. underLevel ile bir alt seviyeye geçilirken level + 1 yapılır; sameLevel ile aynı seviyede kalınarak ilerlenir. Bu da BFS değil DFS tabanlı bir gezinmeye karşılık gelir.

Belirli bir seviyede en fazla çalışana sahip kişiyi bulmak için maxEmployeeForLevel fonksiyonu çalıştırılır. Bu fonksiyon, kullanıcının girdiği seviyede bulunan tüm çalışanları sameLevel üzerinden yatay olarak tarar ve her biri için howManyEmployeeAtEachLevel çağrılarak o kişinin astları sayılır. En fazla ast sayısına sahip kişi maxEmployee olarak belirlenir. Hesaplama yapılırken yalnızca belirtilen seviyenin altındaki çalışanlar dikkate alınır.

Tüm çalışanların yaş ortalamasını hesaplamak için ageAverage fonksiyonu kullanılır. Bu fonksiyon içinde çağrılan accumulateAge, tüm düğümleri dolaşarak toplam yaş ve çalışan sayısını toplar. Özyineleme yoluyla hem alt hem de aynı seviyedeki düğümler

gezilir ve yaşlar toplanır. İşlem sonunda yaş ortalaması (toplam yaş) / (çalışan sayısı) şeklinde elde edilir.

Benzer şekilde maaş giderini hesaplayan salaryExpense fonksiyonu da accumulateSalary isimli özyinelemeli yardımcı fonksiyonu çağırarak tüm çalışanların maaşlarını toplar. underLevel ve sameLevel zincirleri ile tüm düğümler gezilerek toplam maliyet hesaplanır.

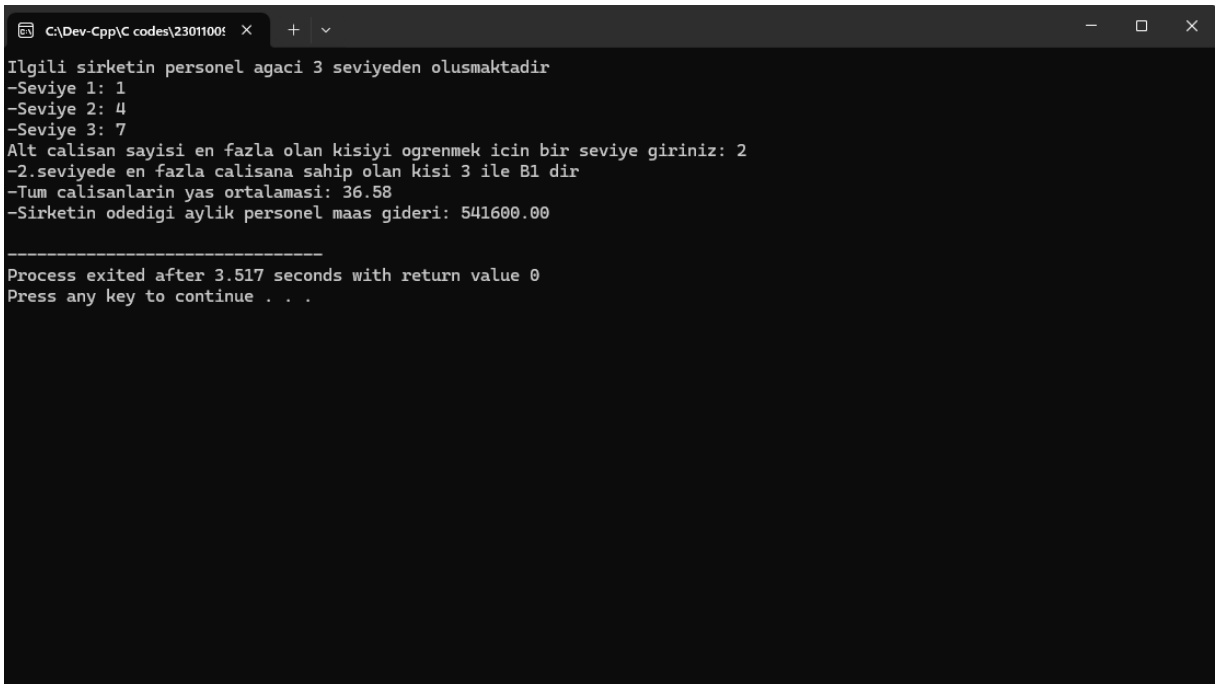
Programın sonunda ise bellekte oluşturulan tüm düğümler freeTree fonksiyonu ile post-order sırayla serbest bırakılır. Önce underLevel, sonra sameLevel alt ağaçları serbest bırakılır ve en son o düğümün kendisi free edilir.

2 – Karşılaşılan Sorunlar:

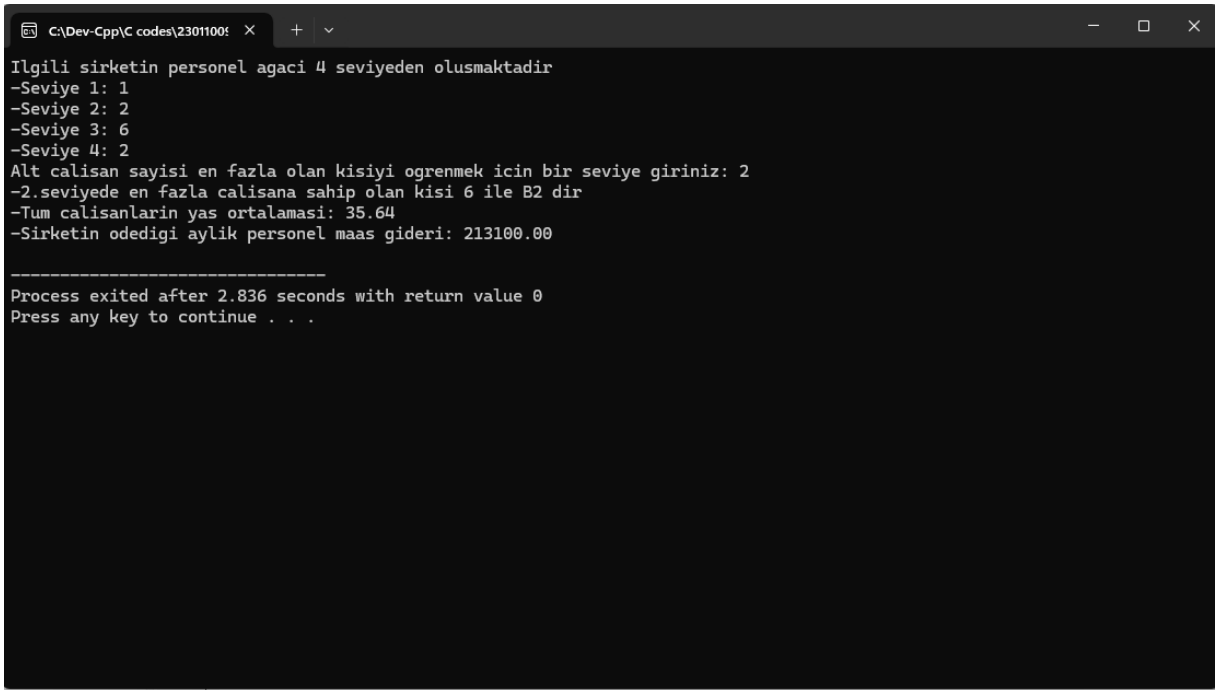
Başlangıçta yaş ortalaması (ageAverage) ve maaş gideri (salaryExpense) hesaplamaları tek bir fonksiyon içerisinde yapılmaya çalışılmıştı. Bu durumda yalnızca kök düğümün (root) yaşı ve maaşı işleme alındığından, sonuçlar tüm çalışanları yansıtmıyordu. Sorunun temelinde özyinelemeli gezinmenin eksik olması yatmaktaydı. Bu problem, toplama işlemini gerçekleştiren accumulateAge ve accumulateSalary isimli yardımcı fonksiyonların ayrı tanımlanmasıyla çözülmüş; böylece tüm ağaç yapısı underLevel ve sameLevel bağlantılarıyla derinlemesine ve yatayına gezilerek doğru hesaplama yapılmıştır.

3-Ekran Çıktıları :

a)Personel.txt



b)Personel2.txt



c)personel3.txt