**1fd75c3872a94e26ad68c7fa7667bdc82c07dd4ba85f4a6793f7a2b4e943b8fd**

**T.C.**

**SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

VERİ YAPILARI DERSİ ÖDEV RAPORU

**VERİ YAPILARI KULLANILARAK BÜYÜK SAYILARIN FAKTÖRİYELİNİN ALINABİLMESİ**

**Grup Elemanları:**

**B161210040 - İbrahim KUŞ**

**B140910040 - Salih Burak DEMİRCİ**

**SAKARYA**

**Kasım, 2017**

Veri Yapıları Dersi

İbrahim KUŞ a[[1]](#footnote-1), Salih Burak DEMİRCİb\*

a B161210040/ I-C Grubu

b B140910040/I-C Grubu

Özet

Büyük sayıların faktöriyellerini hesaplamakta bilgisayarlar belleklerin büyük bölümü bu hesaplama için ayrılır ve belleğin büyük bölümü bu işlem için ayrıldığı için bilgisayar performansında azalma gözlemlenebilir. Bu ödevde bilgisayar bilimlerinde yer alan çeşitli veri yapılarıyla problemi çözmemiz istendi. Bellek ve bellek alt alanlarında işlemler yaparak ve heap bellek bölgesinde uygun olan veri yapılarını kullanarak bellek yönetimini ve maliyeti optimize ettik. Çok büyük sayıların faktöriyelleri bilgisayarlar tarafında çok büyük zaman kaybı ve maliyet oluşturmaktadır. Böyle sayıların faktöriyellerini stack(hafıza)’te tutarak daha büyük sayıların faktöriyellerini hesaplamamızda bize maliyeti optimize etme ve zaman tarafında yarar sağlamaktadır. Bu problemi kendimizin tanımladığı, hesaplamada Sayı sınıfı kullanarak büyük sayıları stack’te tutarak hızlı bir şekilde hesapladık.

© 2017 Sakarya Üniversitesi.

Bu rapor benim özgün çalışmamdır. Faydalanmış olduğum kaynakları içeresinde belirttim. Her hangi bir kopya işleminde sorumluluk bana aittir.

Anahtar Kelimeler: Yığıt, Düğüm, Bağıl Liste, Pointer, Heap Bellek Bölgesi, Büyük Sayıların Faktöriyelleri

1. GELİŞTİRİLEN YAZILIM

Bize verilen problemde C++ üzerinde bellek yönetimiyle ilgili daha kolay işlemler yapabildiğimiz için seçtik. Bu problemde stack, bağıl liste ve düğüm veri yapılarını bir arada kullanarak bize verilen problemi daha kolay hale getirdik. Stack veri yapısını LIFO(Last In First Out) yönteminin bazı avantajları olduğu için kullandık. Kendimizin tanımlamış olduğu Sayı sınıfını ise büyük sayıların hesaplamasında kullandık. Bağıl liste ve düğüm veri yapılarını da bir arada kullanarak bellek üzerinde daha esnek işlemler yapabildiğimiz için tercih ettik. Ödev bellek üzerinde bellek alanını optimize edebileceğimiz ve maliyet konusunda çeşitli veri yapılarının kullanılarak düşük maliyet ile çeşitli sorunlara karşı veri yapılarının işlevselliğini deneyimledik.

1. ÇIKTILAR

Farklı sayıları girdiğimiz zaman girdiğimiz sayıların öncelikle faktöriyellerini hesaplarız ve stack veri yapısında saklarız. Yeni girdiğimiz sayılar, stack’te tuttuğumuz sayıdan küçükse hızlı bir şekilde stack’te tutulan sonucu getirecektir. Eğer stack’te tuttuğumuz sayıdan büyükse, stack’te tutulan sayının faktöriyeli kullanarak yeni sayının faktöriyeli hesaplanır ve yeni değer stack veri yapısına atılarak saklanır.

1. SONUÇ

Çalışmamızdan elde ettiğimiz sonuçlara baktığımızda büyük sayıların faktöriyellerini çeşitli veri yapılarını kullanarak kolayca bulabildiğimizi, veri yapılarını kullanarak bellek yönetimi açısından hem maliyet hem de zaman açısından tasarruf edebildiğimizi ortaya koyduk.

Referanslar

[1] C/C++ ile Veri Yapıları ve Çözümlü Uygulamalar , Prof. Dr. Nejat Yumuşak , Yrd. Doç. Dr. Muhammed Fatih Adak, Mart 2016, Seçkin Yayıncılık, ISBN 9789750237065

[2] Veri Yapıları Ders Notları, Sakarya Üniversitesi

1. \* İbrahim KUŞ--B161210040, Salih Burak Demirci--B140910040,

   [b161210040@sakarya.edu.tr](mailto:b161210040@sakarya.edu.tr), b140910040@sakarya.edu.tr [↑](#footnote-ref-1)