Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Programlama Laboratuvarı 1

Proje 3

Havalimanı Uçuş Yönetim Sistemi

Robera Tadesse GOBOSHO
ID:190201141
robtad318@gmail.com

Muhammad Abdan SYAKURA
ID:200201147
prof.syakur@gmail.com

ÖZET

Bu rapor Programlama Laboratuvarı 1 dersi 3. Projesinin çözümünü açıklamak üzere hazırlanmıştır. Bu proje C diliyle yapılmıştır. Öncelikli kuyruk (priority queue) uygulanarak İstanbul Havalimanı için bir havalimanı uçuş yönetim sistemi geliştirilmiştir. Öncelikle havalimanına inmek için talep eden uçaklar "input.txt" dosyasından okunmaktadır. Dosya içerisinde oncelik_id, ucak_id ve talep_edilen_inis_saati 3 tane giriş bulunmaktadır ve tek tek satırları okunmaktadır. Her satır okunurken talep edilen iniş saatine bakılır ve kontrol edilir. Eğer daha önce aynı saatte başka uçak talep etmişse, iki uçağın öncelik id'lerine karşılaştırılır. Hangisi daha yüksek öncelik id'ye sahip ise o uçak talep edilen saatte iniş yapacaktır. Diğer uçak ise 1 saat ertelenir. Ve ertelendikten sonraki saati de kontrol edilecektir. O saatte aynı uçak varsa aynı işlem görülecektir. Eğer iki uçağın talep edilen saati ve öncelik id'si aynı ise bu durumda uçak id'lerini kontrol edilir. Eğer bir uçak 3 kereden fazla ertelenirse o uçak iniş için Sabiha Gökçen Havalimanı'na yönlendirilir. Ve bir günde sadece 24 tane uçak iniş için izin isteyebilmektedir. Bu işlemler de input dosyasının son satırına kadar devam edecektir. Kalkış için de iniş saatlerinden 1 sonraki saat yapılır. İnişleri düzeldiği için kalkışta saat çakışması söz konusu değildir. Bu işlemler de "output.txt" dosyasına aktarılır. Dosya içerisinde 6 tane bilgi içermektedir: oncelik_id, ucak_id, talep_edilen_inis_saati, inis_saati, gecikme_suresi, kalkis_saati.

GİRİŞ

1 iniş 1 kalkış olmak üzere 2 pisti bulunan İstanbul Havalimanı'nda gün içerisinde (1-24 saat dilimi boyunca) yapılan uçuşların yönetimi için bir sistem geliştirildi. Havalimanında aynı anda sadece 1 uçak kalkış yapabiliyorken sadece 1 uçak iniş yapabilmektedir. Uçakların her biri önceliklere sahiptir ve 1 günde maksimum 24 uçak iniş için izin isteyebilmektedir. Havalimanındaki uçakların öncelik sırası, iniş saati, gecikme süresi ve kalkış saati bilgileri kullanılarak; iniş pistini ve kalkış pistini kullanım sırasının belirlenmektedir.

Bu projede öncelikli kuyruk kullandık. Öncelikle kuyruk, ilk giren eleman ilk çıkar (First In First Out – FIFO) mantığında çalışan bir veri yapısıdır. Örneğin, kuyruk veri yapısı bankada işlem yaptırmak için sıraya girmiş insanlara benzetilebilir. Sıraya ilk giren kişi ilk işlem yaptıracaktır. Kuyruk tasarımı için dizi ya da bağlı liste kullanılabilir. Dizi kullanılan sabit boyutlu, bağlı liste kullanılarak değişken boyutlu kuyruk oluşturulabilir. Kuyrukta işlemler iki uçtan yapılır. Kuyruk veri yapısında yapılabilecek işlemlerden bazıları aşağıdaki gibidir:

- enqueue(): Kuyruğun önüne eleman ekler.
- dequeue(): Kuyruğun sonundan eleman çıkarır.
- peek(): Silme işlemi uygulamadan sıradaki elemanı (front işaretçisinin gösterdiği düğüm) döndürür.

Öncelikli kuyruk ise bazı problemlerin çözümünde doğrudan kuyruk oluşturulamaz. Örneğin; bir hastanede muayene sırasına girmiş insanlar arasında durumu acil olan birisi bulunabilir ve bu kişi muayene için öncelikli hale gelebilir. Bu gibi durumlarda öncelikli kuyruk kullanılır. Öncelikli kuyrukta ilk giren ilk çıkar mantığı geçerli değildir, önemli olan önceliktir. Öncelikli kuyruk veri yapısında yapılabilecek işlemlerden bazıları aşağıdaki gibidir:

- add(): Kuyruğa eleman eklemek için kullanılır.
- poll(): Kuyruktaki son elemanı döndürür ve elemanı kuyruktan siler.
- peek(): Kuyruktaki son elemanı silmeden döndürür.
- clear(): Kuyruktaki bütün elemanları siler.
- remove(): Kuyruktaki belirtilen elemanı siler.

Öncelikli kuyruk, bir dizi, bağlı liste, heap veri yapısı veya ikili arama ağacı kullanılarak uygulanabilir. Bu veri yapıları arasında heap veri yapısı, öncelik kuyruğunu verimli bir şekilde uygulanmasını sağlar. Heap'te bir veri eklemek ve silmek için en kötü durumu O(Log n)'dir. Bundan dolayı bu projede heap veri yapısını kullandık.

YÖNTEM

İlk olarak readInput fonksiyonuyla input dosyasını okunur. Eğer aynı dizinde input dosyası bulunmuyorsa veya dosya açarken bir hata oluyorsa Fig. 1'deki gibi bir mesaj yazılacak. Eğer bir hata yoksa Fig. 2'deki gibi program sorunsuz açılacak. Dosyanın ilk satırında başlık olduğundan ilk satırından uçakların bilgilerine atlanır. Her satır okunduğunda sırasıyla PlaneP'ye atandıktan sonra PlaneP addToLandingList fonksiyonuna gönderilir ve kontrol edilir:

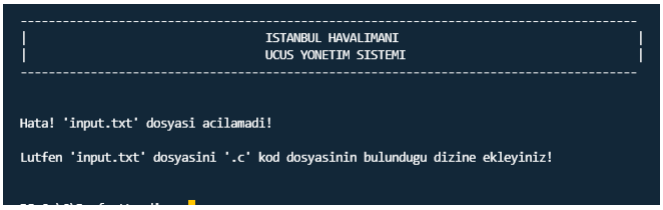


Fig. 1. input dosyası açarken hata varsa

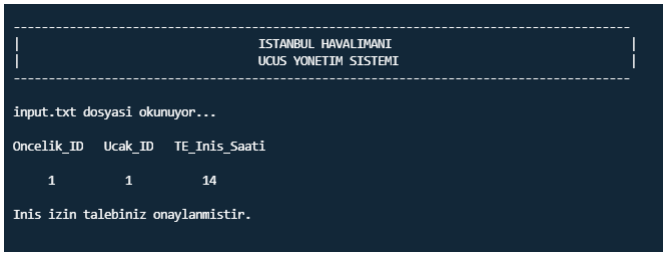


Fig. 2. input dosyası açmayı başardı

- **Dizi boş ise**

Front ile rear 0 olarak tanımlanır ve gönderilen uçak PlaneL'in 0. İndisine atanır. FlagM değişmediği için 1 kalır ve ekrana "İniş izin talebiniz onaylanmıştır." Yazar.

- **Eleman varsa ve dolu değilse**
  Rear 1 artacak. Gönderilen uçak PlaneL rear'inci indisine atanır. Bundan sonra da daha önce aynı saatte iniş talep eden uçakların olup olmadığını delayHandler fonksiyonuyla halledilir. Bu fonksiyon dizideki tüm elemanlarını kontrol eder. Eğer daha önce aynı saatte başka uçak talep etmişse, iki uçağın öncelik id'lerine karşılaştırılır. Hangisi daha yüksek öncelik id'ye sahip ise o uçak talep edilen saatte iniş yapacaktır. Diğer uçak ise 1 saat ertelenir. Ve ertelendikten sonraki saati de kontrol edilecektir. O saatte aynı uçak varsa aynı işlem görülecektir. Eğer iki uçağın talep edilen saati ve öncelik id'si aynı ise bu durumda uçak id'lerini kontrol edilir. Eğer bir uçak 3 kereden fazla ertelenirse o uçak iniş için Sabiha Gökçen Havalimanı'na yönlendirilir. Ekrana bir mesaj yazar.
- **Dizi dolu ise**
  Ekrana "Inis pisti kullanim sirasi dolmustur! Acil inis kosullari kontrol ediliyor... Lutfen bekleyin..." yazar. Gönderilen uçağı en son indise atanır. Çünkü dizi MAX değerinden 1 tane daha fazla yer tanımlanmıştır. Son olarak da delayHandler4Full fonksiyonu çağırılır kontrol edilir. Priority id'si daha yüksek ise acil iniş için izin verilecektir.

Eğer uçak iniş için izin verilirse FlagM değişmez 1 kalır ve ekrana "İniş izin talebiniz onaylanmıştır." Yazar. Sonra da güncel kalkış pisti kullanım sırası ekrana yazdırılacaktır.

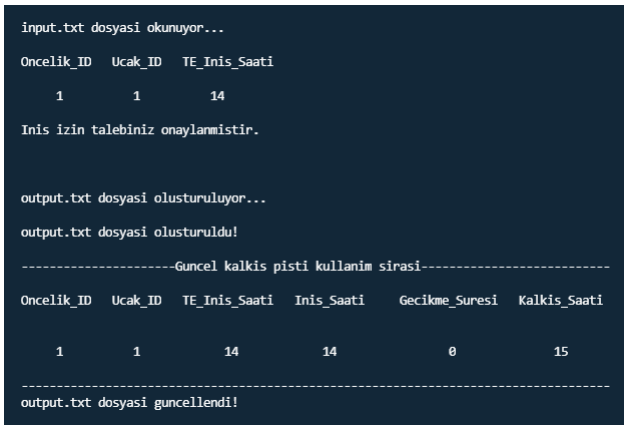


Fig. 3. ilk uçak girdi

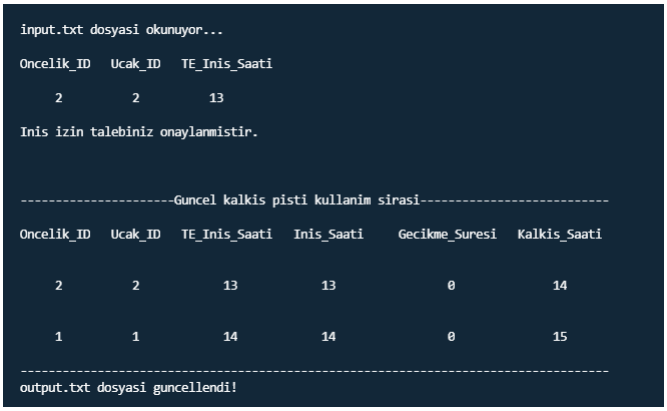


Fig. 4. ikinci uçak girdi

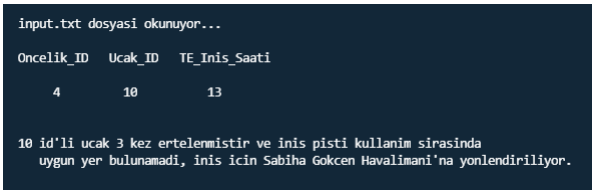


Fig. 5. uçak 3 kez ertelenir ve iniş için uygun yer bulunamayıp Sabiha Gokcen Havalimani'na yönlendirilir.

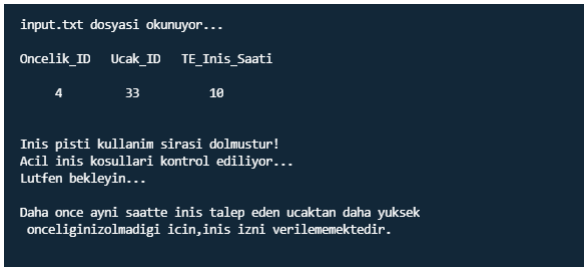


Fig. 6. dizi doluysa ve gelen uçağın daha DÜŞÜK priority id'ye sahip olursa

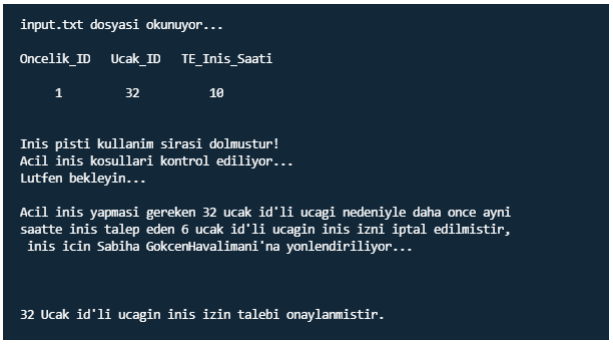


Fig. 7. dizi doluysa ve gelen uçağın daha YÜKSEK priority id'ye sahip olursa

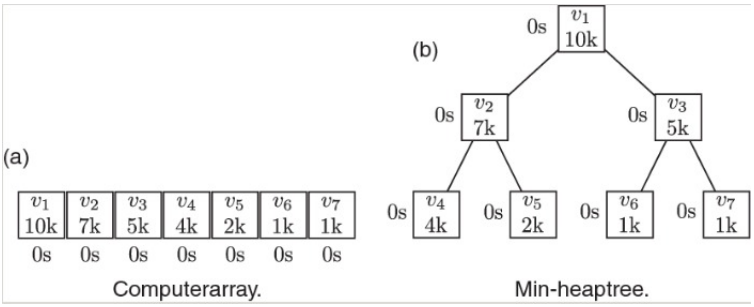
YALANCIKOD (PSEUDOCODES)

```
1  Procedure Min-Heapify(V, i)
       Input: V: an array of elements; i: an index array.
       Output: V modified such that element i roots a min-heap
       // Setting left and right child of node V[i]
2      l ← Left(i);
3      r ← Right(i);
       // Checking for the smallest element
4      if l ≤ |V| and V[l] < V[i] then
5          min ← l;
6      else
7          min ← i;
8      if r ≤ |V| and V[r] < V[min] then
9          min ← r;
       // Updating the min-heap tree
10     if min ≠ i then
11         Exchange V[i] and V[min];
12         Min-Heapify (V, min);
```

```
1  Procedure MHSA(V, P)
       Input: V: Computer array, sorted by capacity in decreasing order; P: set
              (queue) of process.
       Output: A scheduling of set P over set V.
2      Build-Min-Heap (V);
3      while P is not empty do
4          Dequeue process p_i;
5          Assign process p_i to computer v_root located at min-heap root;
6          Update the load of computer v_root;
7          Min-Heapify (V, root);
```

DENEYSEL SONUÇLAR

Havalimanına iniş yapacak uçaklar öncelikle kuleden iniş yapabilmek için



izin talep eder. İniş izni talep eden her bir uçak için havalimanında yeterli kapasite olup olmadığı kontrol edilir. (inis_pisti_kullanim_sirasi öncelikli kuyruğunda yeni uçak eklemek için boş alan var mı?). Kuleden iniş izni talep eden uçaklar için öncelikle, iniş talep edilen saatte pistin dolu mu boş mu olduğu kontrol edilir. Pist boş ise iniş yapılmak istenen saate izin verilir ve inis_pisti_kullanim_sirasi'nda uygun yere eklenir. Aksi halde uçakların iniş sıralaması önceliğe göre belirlenir. Uçakların öncelik (oncelik_id) sıralaması şu şekildedir (yüksekten düşüğe):

1) Ambulans uçağı
2) Savaş uçağı
3) Yolcu uçağı
4) Kargo uçağı

Tüm uçakların iniş ve kalkış süreleri eşittir ve hesaplamalara dâhil edilmeyecektir. Havalimanına iniş yapan her uçağın, kalkış için bekleme süresi 1 saattir. Uçakların kalkış saatine, ötelenmeden dolayı oluşan gecikme süreleri dâhil edilir. Kalkış saati bu bilgiler göz önünde bulundurularak hesaplanmıştır. Kuleden bir günde maksimum 24 uçak iniş için izin talep edebilir. Eğer bu kapasite dolmuşsa; o İniş için onay alan uçaklardan en az birinin önceliği (X uçağı olsun), iniş izni onayı bekleyen uçağın (Y uçağı olsun) önceliğinden düşükse; yüksek öncelikli yeni

uçağa (Y) iniş onayı verilir. Daha önce onay almış ve önceliği düşük olan uçak (X) başka bir havalimanına yönlendirilecektir. O İniş izni daha önceden onaylanan uçağın (X) izni iptal edilmişse; “Acil iniş yapması gereken (Y) no’lu uçağı nedeniyle daha önce aynı saatte iniş talep eden (X) no’lu iniş izniniz iptal edilmiştir, iniş için Sabiha Gökçen Havalimanı’na yönlendiriliyor.” şeklinde ekranda yazdırılacaktır. İniş izni talep eden uçakların her biri satır satır input.txt’den okunur. Okunan her bir satır ekranda gösterilmektedir. Her yeni input satırı okunduğunda, kalkış yapacak olan uçakların bulunduğu output.txt dosyası güncellenir ve güncel kalkis_pisti_kullanim_sirasi öncelikli kuyruğu ekranda gösterilir. output.txt dosyası uçakların kalkış saati bilgisine göre oluşturulmaktadır. Dosyanın içeriği şu şekildedir:

- Her satırda oncelik_id, ucak_id, talep_edilen_inis_saati, inis_saati, gecikme_suresi, kalkis_saati olmak üzere toplamda 6 bilgi içermektedir.
- inis_saati bilgisi; eğer uçak talep ettiği saatte iniş yapmışsa talep_edilen_inis_saati ile aynıdır. Ancak talep edilen saatte önceliği yüksek başka bir uçak iniş yapacaksa belirlenen yeni iniş saati, kalkış saatinin hesaplanmasında kullanılmaktadır.
- Bilgiler input.txt’de olduğu gibi boşluk ile ayrılmaktadır.

Projede kullanılmış olan veri yapıları aşağıdaki gibidir:

- PlaneL (inis_pisti_kullanim_sirasi): Havalimanına iniş yapacak uçakların iniş pistini kullanım sırasını içeren öncelikli kuyruktur. Uçak öncelik ve iniş saati bilgisi dikkate alınarak hazırlanmıştır.
- PlaneT (kalkis_pisti_kullanim_sirasi): Havalimanından kalkış yapacak uçakların kalkış pistini kullanım sırasını içeren öncelikli kuyruktur. Uçak öncelik ve kalkış saati bilgisine göre hazırlanmıştır.

----------Guncel kalkis pisti kullanim sirasi----------

Oncelik_ID	Ucak_ID	TE_Inis_Saati	Inis_Saati	Gecikme_Suresi	Kalkis_Saati
1	11	1	1	0	2
1	12	1	2	1	3
2	23	2	3	1	4
3	17	1	4	3	5
3	7	2	5	3	6
4	28	3	6	3	7
3	24	8	8	0	9
2	13	9	9	0	10
2	6	10	10	0	11

Fig. 8. Güncel kalkış pisti kullanım süresi

2	13	9	9	0	10
2	6	10	10	0	11
2	3	11	11	0	12
3	4	10	12	2	13
2	2	13	13	0	14
1	1	14	14	0	15
3	8	14	15	1	16
1	25	16	16	0	17
3	14	16	17	1	18
4	5	16	18	2	19
3	21	19	19	0	20

Fig. 9.

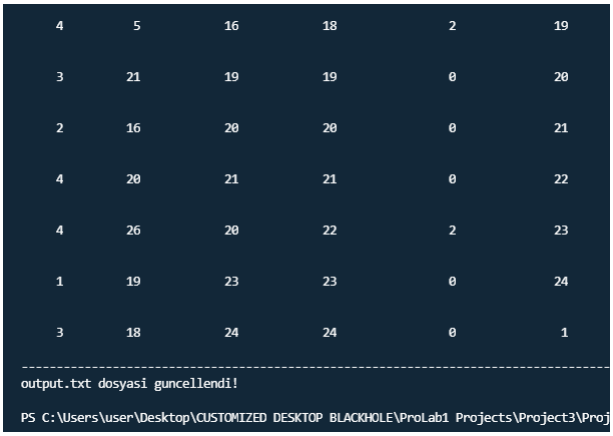


Fig. 10.

KAYNAKÇA

REFERENCES

[1] https://medium.com/@tolgahan.cepel/do%C4%9Frusal-veri-yap%C4%B1lar%C4%B1-4-kuyruk-queue-dcbd07e8ba77
[2] https://www.geeksforgeeks.org/priority-queue-set-1-introduction/
[3] https://www.programiz.com/dsa/priority-queue
[4] https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html
[5] https://www.programiz.com/dsa/heap-data-structure
[6] https://www.geeksforgeeks.org/binary-heap/
[7] https://www.sanfoundry.com/c-program-implement-heap/
[8] https://www.youtube.com/watch?v=HqPJF2L5h9Ut=1335s
[9] https://www.youtube.com/watch?v=wptevk0bshY
[10] https://www.youtube.com/watch?v=_U1AJZQxYTU
[11] https://www.codesansar.com/c-programming/recursive-function.htm
[12] https://www.tutorialspoint.com/cprogramming/c_recursion.htm
[13] https://www.programiz.com/c-programming/c-recursion
[14] https://www.youtube.com/watch?v=kepBmgvWNDw
[15] Book Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology by Dan Gusfield
[16] https://www.youtube.com/watch?v=ngCos392W4w
[17] https://www.youtube.com/watch?v=ggk7HbcnLG8
[18] https://www.youtube.com/watch?v=QWDCKPEWSWot=425s
[19] https://www.youtube.com/watch?v=cv7CY8UmFL0
[20] https://www.geeksforgeeks.org/readwrite-structure-file-c/
[21] https://cboard.cprogramming.com/c-programming/77351-writing-array-struct-file.html
[22] https://courses.cs.washington.edu/courses/cse373/17wi/lectures/priority-queues.pdf
[23] http://www.cs.cornell.edu/courses/cs2110/2015fa/L17-PriorityQueuesAndHeaps/cs2110PqueuesHeaps.pdf
[24] https://www.geeksforgeeks.org/priority-queue-using-binary-heap/
[25] https://runestone.academy/runestone/books/published/pythonds/Trees/BinaryHeapImplementation.html
[26] https://www.hackerearth.com/practice/notes/heaps-and-priority-queues/
[27] https://algs4.cs.princeton.edu/24pq/
[28] https://cgi.csc.liv.ac.uk/ leszek/COMP202/week4/week4.pdf
[29] https://www.overleaf.com/