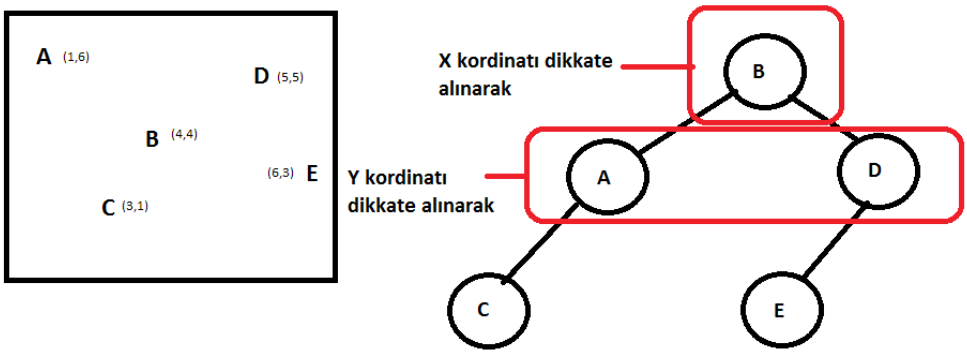
FIT 2 [INF224] – 2019/20

Veri Yapilari ve Algoritmalar – TP 05

Gulfem Isiklar Alptekin – Ozgun Pinarer

Amac

k-d ağaçları ikili arama ağaçlarının özelleşmiş bir türüdür. Ağacın her bir elemanı, o elemanın xy koordinatlar sistemindeki yerini gösteren iki tane sayı içerir. Ağaç kurulurken her bir levelda x ya da y değerlerine göre elemanlar yerleştirilir. Daha açıkça ifade etmek gerekirse, root olarak alınan node'dan sonra, 2. Levelda eğer eklenecek node'un x değeri roottan küçükse sola, büyükse sağa eklenir. Bir sonraki levelda tekrar y'e bakılır. Bu şekilde her node ağaca eklenerek ağaç kurulur.



En yakın komşu Problemi (nearest neighbor problem)

x ve y kordinatları düzleminde tüm noktalar x ve y olmak üzere iki kordinat değerine sahiptir. Birinci noktanın koordinatları x1,y1 ikinci noktanınkiler ise x2,y2 olmak üzere iki nokta arasındaki en yakın uzunluk kök içerisinde $\sqrt{(x2-x1)^2+(y2-y1)^2}$ olarak hesaplanır. En yakın komşu problemi, verilen bir noktalar kümesi için, yine verilen bir noktaya en yakın olan komşu noktasını bulma problemidir. K-d ağaçları sayesinde tüm noktaların aradığımız noktaya olan uzaklıklarını bu formülle hesaplayıp sıralama derdinden kurtulmuş oluyoruz. K-d ağacında x ve y koordinat değerlerine göre alandaki noktaları parçalara bölerek bir ağaç oluşturuyoruz.

Kapalıçarşı'da yol bulma

Bu çalışmada kapalı çarşının sayısız kapısı arasında kaybolan turistler için en yakındaki kapıyı bulmalarını sağlayacak bir yöntem geliştireceğiz. Aşağıda kapalı çarşının haritasını göreceksiniz. Bu haritada kapılar kırmızı renkli harflerle belirtilmiştir. Haritanın altındaki tabloda her kapının index harfi, ismi, x ve y koordinatının listesini bulacaksınız. Kapalı çarşının merkezi kabul edilen Cevahir Bedesteni de haritada T harfi ile gösterilmiştir.

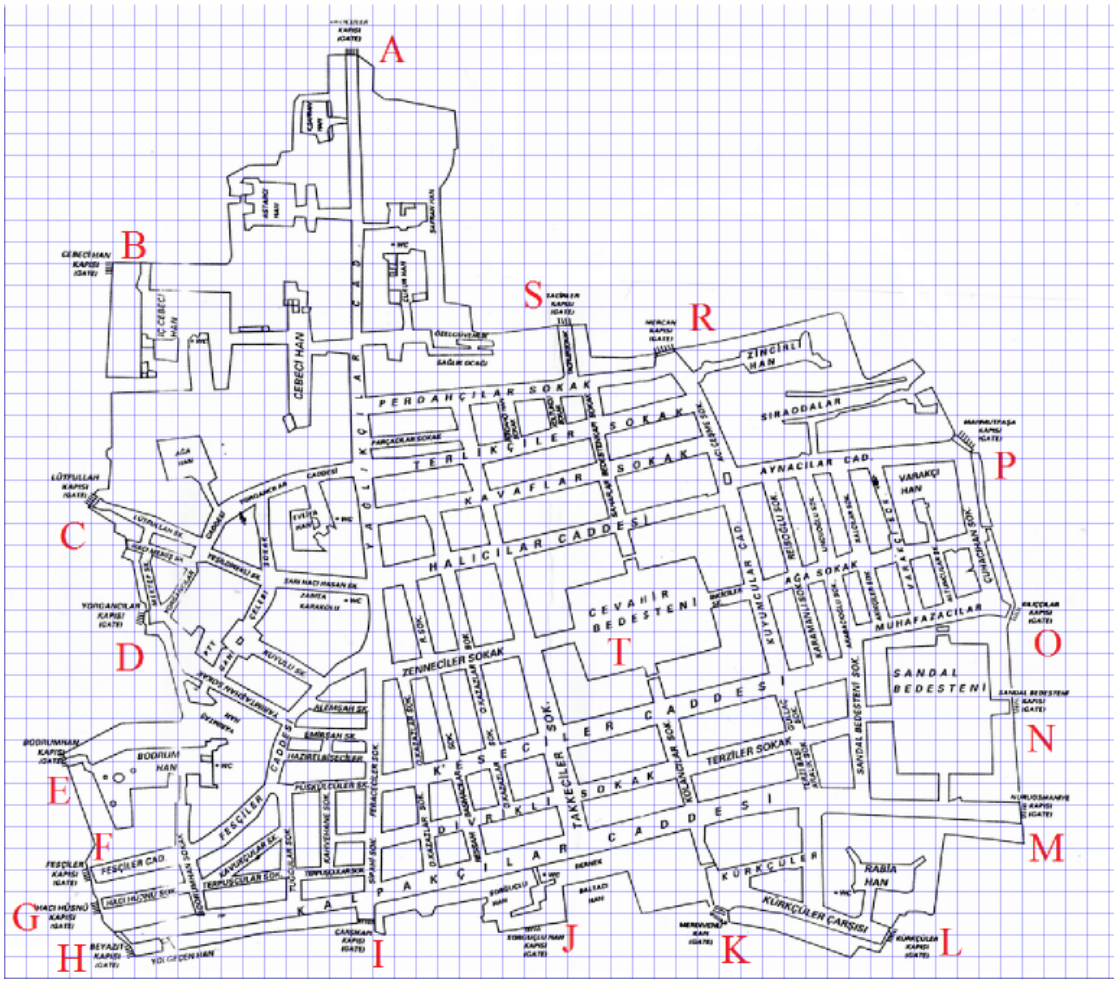


Figure 1. Kapalı çarşının haritası. Kırmızı Harfler kapıları göstermektedir

Table 1. Koordinatlar ve Kapı İsimleri

İndex	Kapı Adı	X koordinatı	Y koordinatı
A	Örücüler	16	43
B	Cebeci Han	5	33
C	Lütfullah	4	22
D	Yorgancılar	6	17
E	Bodrum Han	3	10
F	Fesçiler	4	5
G	Hacı Hüsnü	4	4
H	Beyazıt	5	2
I	Çarşıkapı	16	3
J	Sorguçlu Han	25	2
K	Merdivenli	32	3
L	Kürkçüler	39	2
M	Nuruosmaniye	46	7
N	Sandal Bedesteni	45	13
O	Kilitçiler	45	17
P	MahmutPaşa	44	24
R	Mercan	32	29
S	Tacirler	26	31
T	CEVAHİR BEDESTENİ - MERKEZ	28	16

SORU 1

Sizce kuracağınız k-d ağacında kapıları ağaca eklerken öncelikle root olarak hangi kapıdan başlamalısınız? Cevabınızı argümanlarınızı açıklayarak belirtiniz.

SORU 2

Herbir kapıyı kodlayabilmek için bir node struct'ı (yapısı) tanımlayınız. Bu yapı, kapı için, o kapının harf indexi, kapının adı, kapının x koordinatı ve kapının y koordinatını içermelidir.

SORU 3

Yukarıda verilen T noktasını yazdığınız node struct'ı cinsinden yaratıp root olarak k-d ağacını kuran create_tree() fonksiyonunu yazınız. Bu fonksiyonu ilerde başka bir noktayı root tanımlayarak tekrar kullanabilirsiniz. Bu nedenle T ile ilgili verileri fonksiyona parametre olarak veriniz.

SORU 4

K-d ağacının her levelına node eklemesi yapacak insert()fonksiyonunu yazınız. Burada level numarası tek ise x koordinatına göre, çift ise y koordinatına göre ekleme yapılacaktır. insert()fonksiyonunu yazabilmeniz için ağacın levelını döndüren level() fonksiyonunu yazmanız gerekmektedir. Level fonksiyonunda kullanılacak is_leaf()fonksiyonunu da yazarak faydalanabilirsiniz.

SORU 5

Soru 3 ve 4'te yazdığınız fonksiyonları kullanarak bir init_tree() fonksiyonu yazınız. Bu tree hem root node ile ağacı yaratacak hem de bu ağaca yukarıda verilen her kapıyı ekleyecek ve ağacın kurulumunu tamamlayacaktır. Kapı ekleme sırası olarak A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, R, S sırasını kullanınız.

SORU 6

Yarattığınız ağaçtan girilen kapı adına göre arama yapıp ilgili node'u bulan search_port() fonksiyonunu yazınız.

SORU 7

Yarattığınız ağaçtan girilen kapı indexine göre arama yapıp ilgili node'u bulan search_index() fonksiyonunu yazınız.

SORU 8

Yarattığınız ağacı inorder'a göre gezecek (travers) edecek ve sonucu ekranda gösterecek print_inorder() fonksiyonunu yazınız.

SORU 9

Yarattığınız ağacı preorder'a göre gezecek (travers) edecek ve sonucu ekranda gösterecek print_preorder() fonksiyonunu yazınız.

SORU 10

Yarattığınız ağacı postorder'a göre gezecek (travers) edecek ve sonucu ekranda gösterecek print_postorder() fonksiyonunu yazınız.

SORU 11

Yarattığınız ağaç dengede(balanced) midir?

Yeni ve dengede bir ağaç yaratmak için önce yarattığınız ağacı öldürün. Yeni ağaç için root node J kapısı olsun. Sırasıyla E, T, H, I, D, A, B, C, R, L, S, P, O, K, M, G, F, N nodelarını ağacınıza ekleyin. Yeni oluşturduğunuz ağacı bir kağıda çizerek dengede olup olmadığını gözlemleyiniz. Kagidin fotografini cekip odev dosyaniza ekleyiniz.

SORU 12

Yarattığınız ağaçtan girilen konum koordinatlarına göre arama yapıp en yakın node'u bulan search_coordinate() fonksiyonunu yazınız.

SORU 13

Yazdığınız kod ile, Terlikçiler Sokağında (25,25) koordinatlarında kaybolan bir turiste yardım etmek için en yakın kapıya (ya da kapılar) onu yönlendirin.

Yazdığınız kod ile, İç Cebeci Han'da (14,34) koordinatlarında alışveriş yapan bir turiste en yakın kapıyı (ya da kapılar) bildirin

Yazdığınız kod ile, Hatip Emin Hanı (5,27) koordinatlarında alışveriş yapan bir turiste en yakın kapıyı (ya da kapılar) bildirin.

SORU 14

Yukarıda yazdığınız print_preorder(),print_inorder() ve print_postorder() fonksiyonlarını kullanarak yeni yarattığınız ağaca göre kapıları tek tek geziniz.

Sizce bu travers yöntemleri ile bir turist her kapıyı çok yorulmadan gezebilir mi? Her defasında bir sonraki en yakın kapıya yönlendirecek bir travers algoritması kurgulayınız. Bu algoritmanın pseudo-code'u ve C kodunu yazarak sonucu ekrana basınız. Son yarattığınız ağaca göre, bu algoritma eğer root'tan başlarsa, travers sırası kapılara göre şöyle olmalıdır:

J – I –H – G – F –E –D –C –B – A –S –R –P –O –N –M –L –K –T