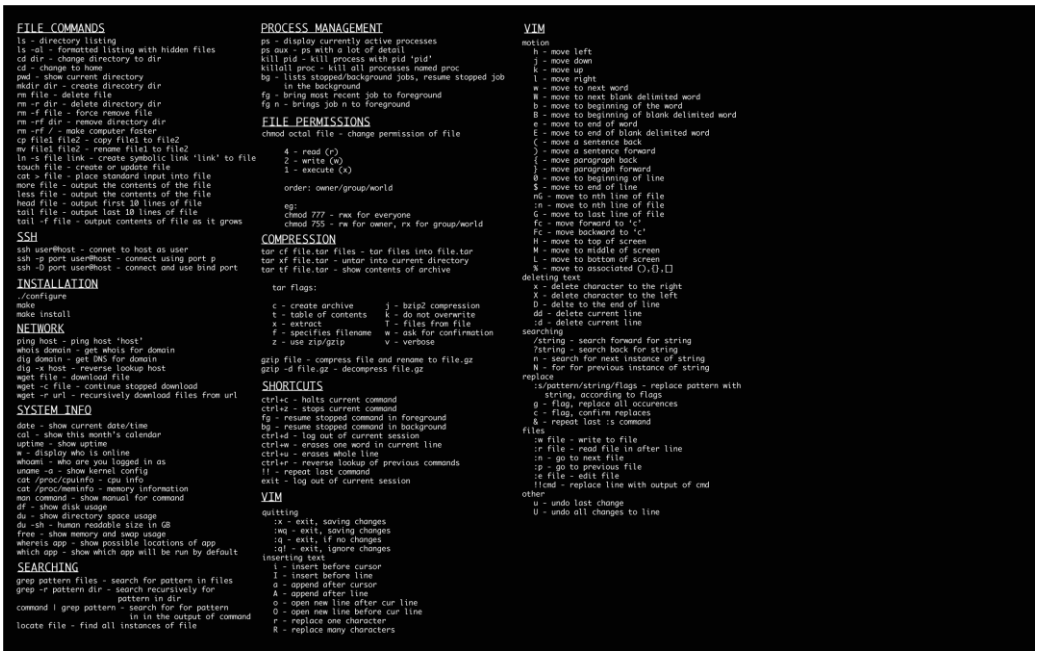
ALGORİTMA ANALİZİ – ÖDEV 1 RAPOR



Closest Points with Divide and Conquer

ELİF YAĞMUR DURAN – 18011071

ALGORİTMA ANALİZİ – ÖDEV 1 RAPOR

CLOSEST POİNTS WİTH DİVİDE AND CONQUER

KULLANILAN FONKSİYONLAR

float bruteforce(int size, int mat[][2]);	N<=3 olduğunda girilen kod bloğunun içerisinde brute force yöntemiyle en yakın noktaları arayan fonksiyon
float distance(int x1, int y1, int x2, int y2);	Kod karışıklığını önlemek amacıyla iki nokta arası uzaklığı float olarak döndüren fonksiyon
float min(float a, float b);	Kod karışıklığını önlemek amacıyla iki argümandan minimum olanını döndüren fonksiyon
float stripcalc(int n, int mat[][2], float d);	Closestfunction içerisinde left ve right taraflar arasındaki minimum uzaklıktaki fonksiyonlar bulunduktan sonra medyana d uzaklığından daha yakın olan noktalar arasında minimum uzaklıktaki noktaları bulan fonksiyon
float closestfunction(int n, int mat[][2]);	Yukarıdaki fonksiyonları içinde barındıran en yakın noktaları bulma rekürsif fonksiyonu
void merge(int mat[][2], int l, int m, int r); void mergesortbyx(int mat[][2], int l, int r);	Matriste x koordinatına göre sort yapmaya ayarlanmış mergesort fonksiyonları

DEĞİŞKENLER

int i, j, k;	Loop sayısı tutmaya yarayan değişkenler
int n;	Nokta sayısı, ayrıca noktaları içinde barındıracak matrisin satır sayısı

int *xptr1, *yptr1, *xptr2, *yptr2;
int x1 = 0; int y1 = 0; int x2 = 0; int y2 = 0;
xptr1 = &x1; yptr1 = &y1; xptr2 = &x2; yptr2 = &y2;
	Global olarak tanımlanmış, fonksiyonlardan birden fazla değer döndürebilmeye yaramak amacıyla pointer olarak tanımlanmış, minimum uzaklıkta olacak noktaları tutma değişkenleri ve onların main içinde initialization ı

FILE *fptr; char str[50];	File işlemleri için kullanılan değişkenler, pointer l ve kullanıcıdan file ismini alan string
double dmin; float temp;	Sqrt fonksiyonu double olmayan türlerde sorun yaratıyor, ancak işlemler float üzerinden yürüyor. Minimum uzaklık en son aşamada float cinsinde olacaktır, daha sonra bu float double a çevrilip

	sqrt fonksiyonuna koyulur, bu yüzden iki değişkene ihtiyacımız vardı.
char selection = 'y';	Programa ne kadar kez devam edileceğini kontrol eden değişken, yes ile başlatılır, kullanıcı no girebilir.

Kodun Açıklaması

BRUTE FORCE FONKSİYONU

N <=3 aşamasına gelindiğinde girilen kod bloğuna ait fonksiyon.

int i = 0;
int j = 0;
int k = 0;
float dist = FLT_MAX;
	Fonksiyona girişte değişken tanımları. Standard 0 dan başlatılan i, j ve k nın dışında birde en kısa mesafe karşılaştırmaları yapmak için temp değişken. Bir float un alabileceği en yüksek değerde başlatılır.

for (i = 0; i < size; i++) {
for (j = (i + 1); j < size; j++) {
if (dist != min(dist,
distance(mat[i][0], mat[i][1], mat[j][0], mat[j][1])))
{
*xptr1 = mat[i][0];
*yptr1 = mat[i][1];
*xptr2 = mat[j][0];
*yptr2 = mat[j][1];
dist = distance(mat[i][0], mat[i][1],
mat[j][0], mat[j][1]);
}
}
}
	i değeri 0 dan başlar, j değeri 1 den ve her i için ondan sonra gelen satırlar kendi satırı ile karşılaştırılır. Böylece her satır, yani her nokta, tekrar yapılmaksızın kendisinden başka her noktayla karşılaştırılmış olur.
	Bu iki katlı loop un içerisindeki if bloğunda da elimizdeki anlık minimum uzaklık değerini tutan dist değeri o anda i ve j satırlarında bulunan iki nokta arası mesafeyle karşılaştırılır (distance ve min fonksiyonları yardımıyla).
	Min fonksiyonundan dönen sonuç dist ten farklıysa yeni bir minimum elde edilmiştir. Noktaları globalde tutan pointerlar sayesinde minimum noktalara ve minimum uzaklığa güncelleme yapılır

return dist;	Bulunan en kısa mesafe döndürülür.

DİSTANCE VE MİN FONKSİYONLARI

Aşağıdaki kod blokları kodun daha kolay anlaşılabilir olmasını sağlamak amacıyla fonksiyonlar içine yazıldı.

float distance(int x1, int y1, int x2, int y2) {	Normalde aldığı noktaların karekökünü döndürmesi gerekir. Ancak sqrt un değişken tipleri

return ((x1 - x2) * (x1 - x2)) + ((y1 - y2) * (y1 - y2)); }	konusunda çıkardığı sorunlar yüzünden, son aşamaya kadar koordinatların sadece kareleri toplamları ile karşılaştırmalar yapan bir distance fonksiyonu tercih edildi. Karekök hesabı main fonksiyonu içinde yapıldı.

float min(float a, float b) {
if (a < b)
return a;
else
return b;
}
	Aldığı argümanlar arasından minimum olanını döndüren fonksiyon.

STRİPCALC FONKSİYONU

Closestfunction rekürsif çalışırken, sağ ve sol tarafların brute force ile aranması tamamlandığında, iki farklı tarafta bulunabilecek olan noktaların uzaklıklarını değerlendirmeye katmak amaçlı fonksiyon.

float stripcalc(int n, int mat[][2], float d) {	Aynı closestfunction gibi üzerinde işlem yapılan matrisi ve size ı temsil eden n i alır, ancak onun aksine birde şimdiye kadar bulunmuş en kısa uzaklığı da alır.

int matstrip[n][2];
float stripd = FLT_MAX;
int i = 0;
int j = 0;
int k = 0;
int mid = n / 2;
	Fonksiyonun içerisinde geçici bir matris oluşturulur, ayrıca brute force ta yapıldığı gibi anlık en kısa mesafeyi tutacak bir geçici değişkende tanımlanır ve en yüksek float değerinden başlatılır.
	Medyan değeri olması için mid değişkenine n / 2 değeri atanır. İnt tipinde olduğu için medyan n / 2 her zaman tamsayı çıkacaktır.

for (i = 0; i < n; i++) {
if ((abs(mat[i][0]) - mid) < d) {
matstrip[j][0] = mat[i][0];
matstrip[j][1] = mat[i][1];
j++;
}
}
	Fonksiyona gelmiş olan bütün noktalar tek tek aranır. Medyana d değerinden daha yakın olanlar geçici matrise alınır. Bu sırada j değişkeni hem geçici matriste bulunulan satırı tutmak için hem de looptan çıkıldığında satır sayısını bilebilmek için kullanılır.

for (i = 0; i < j; i++) {
for (k = (i + 1); k < j; k++) {
if (d != min(d, distance(matstrip[i][0], matstrip[i][1], matstrip[k][0], matstrip[k][1]))) {
	Sıradaki loopta artık matstrip geçici matrisi üzerindeki en kısa mesafe aranmaya başlanır. i ve k brute force ta kullanılan yaklaşımın aynısı amaçla iki farklı satırdaki noktaları karşılaştırmak için kullanılır. j matstrip in büyüklüğüdür. Bu looplar

*xptr1 = matstrip[i][0];
*yptr1 = matstrip[i][1];
*xptr2 = matstrip[k][0];
*yptr2 = matstrip[k][1];
d = distance(mat[i][0], mat[i][1], mat[k][0], mat[k][1]);
}
}
}
	içerisinde fonksiyona girmeden önce bilinen d değerinden daha kısa bir mesafeye rastlanırsa bu stripd de tutulur ve noktalar güncellenir.

return stripd;}	Stripd döndürülür.

CLOSESTFUNCTİON FONSKİYONU

Rekürsif çalışan, programın ana amacını yerine getiren fonksiyon.

float closestfunction(int n, int mat[][2]) {	Üzerinde işlem yapılacak matrisi ve matrisin satır sayısını alır. Zaten iki sütun olacağı belli olduğu için başka bilgiye ihtiyaç olmaz.
float d; float dl, dr;	Minimum adayını tutacak d değişkeni ile sağ ve sol taraftan döndürülecek olan minimum adaylarının karşılaştırılması için geçici diğer iki değişken.

if (n <= 3) {
d = bruteforce(n, mat);
}
	N 3ten küçük olduğunda brute force fonksiyonuna girer.

else {
int mid = n / 2; //mid is the row number
 	
dl = closestfunction(mid, mat);
dr = closestfunction((n - mid), mat + mid);
 	
d = min(dl, dr);
d = min(d, stripcalc(n, mat, d));
}
	Eğer n 3ten büyük ise medyan hesaplanır.
	Önce matrisin medyana kadar olan yarısı (sol yarı) rekürsif olarak hesaplanır ve dl e döndürülür. Daha sonra aynısı sağ yarı için yapılır. Bunu temsil etmek için pointer aritmetiği ile matrisin medyandan sonraki üyelerinin fonksiyona yollanması sağlanmıştır.
	Daha sonra dl ve dr arasından küçük olan d ye seçilir. Ve d stripcalc ten dönen sonuç ile karşılaştırılır.

return d;}	En son kalan d döndürülür.

MERGESORT FONKSİYONLARI

Mergesort un bir matrisi x koordinatlarına göre sort etmesine uyarlanmış hali.

void merge(int mat[][2], int l, int m, int r) {
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;
 	
int L[n1][2], R[n2][2];
	Sağ ve sol matrislerin uzunlukları hesaplanır ve temp amaçlı yeni sağ ve sol matrisler initialize edilir.

for (i = 0; i < n1; i++) {
L[i][0] = mat[l + i][0];
L[i][1] = mat[l + i][1];
}
 	
for (j = 0; j < n2; j++) {
R[j][0] = mat[m + 1 + j][0];
R[j][1] = mat[m + 1 + j][1];
}
	Orijinal matristeki değerler geçici sağ ve sol matrislere atanır.

i = 0;
j = 0;
k = l;
while (i < n1 && j < n2) {
if (L[i][0] <= R[j][0]) {
mat[k][0] = L[i][0];
mat[k][1] = L[i][1];
i++;
} else {
mat[k][0] = R[j][0];
mat[k][1] = R[j][1];
j++;
}
k++;
}
	Sağ taraftaki ve sol taraftaki değerler sırasıyla okunur. K bu yüzden sol matrisin büyüklüğü değerinden başlatılmıştır.
	Sıradaki iki değerden küçük olanı orijinal matriste sıraya yerleştirilir.

while (i < n1) {
mat[k][0] = L[i][0];
mat[k][1] = L[i][1];
i++;
k++;
}
	Üstteki loop sağ veya soldan herhangi birinde eleman kalmadığında biteceği için herhangi birinde geride kalan elementler direk ana diziye bu while lar ile yazılır.

while (j < n2) {
mat[k][0] = R[j][0];
mat[k][1] = R[j][1];
j++;
k++;
}
}	

void mergesortbyx(int mat[][2], int l, int r) {
if (l < r) {
int m = l + (r - l) / 2;
//
mergesortbyx(mat, l, m);
mergesortbyx(mat, m + 1, r);
//
merge(mat, l, m, r);
}
}
	Standart mergesort fonksiyonu.
	Right left i geçmediği sürece ikiye bölerek devam eder. Middle right ve left in ortasındaki sayıdır.
	Önce sol taraflar çağırılır, daha sonra sol taraflar. Bu ikisi bittikten sonra iki taraf yukarıda açıklandığı gibi merge edilir.

MAİN

int main() içinde olanlar.

while (selection == 'y') {
printf("please enter sample file name:");
scanf("%s", &str);
printf("how many points are there?\t");
scanf("%d", &n);
int mat[n][2];
fptr = fopen(str, "r");
	Programın seçim y olduğu sürece tekrar çalışması sağlanmıştır.
	Kullanıcı file ismini ve kaç tane nokta olduğunu girer.
	Sonra noktaları dosyadan alıp gerekli işlemler için saklayacak olan matris oluşturulur ve file pointer ı file ı açar.

if (fptr == NULL) {
printf("file failed to open.");
}
	Filepointer ı null e eşitse kullanıcıya uyarı verir.

else { printf("file has now opened.\nexecuting program...\n");	File açılabildiyse program devam eder.
//prep matrix here for (i = 0; i < n; i++) {	File dan noktalar okunup matrise aktarılır.

for (j = 0; j < 2; j++) {
int a;
fscanf(fptr, "%d", &a);
mat[i][j] = a;
}
}	

printf("matrix is loaded.\n");
for (i = 0; i < n; i++) {
printf("%d %d\n", mat[i][0], mat[i][1]);
}
printf("sorting the matrix...\n");
//merge sort here
mergesortbyx(mat, 0, n - 1);
printf("sorted matrix:\n");
for (i = 0; i < n; i++) {
printf("%d %d\n", mat[i][0], mat[i][1]);
}
	Dosyanın düzgün okunduğu ve matrise noktaların yüklendiği gösterilir.
	Daha sonra mergesort yapılır ve matrisin sort edilmiş hali de gösterilir.

printf("now calculating closest points...\n");
temp = closestfunction(10, mat);
dmin = sqrt(temp);
printf("closest points are: (%d, %d) and (%d, %d) + the smallest distance is -> %f", x1, y1, x2, y2, dmin);
	Artık program asıl görevini yapmaya geçer.
	Closestfunction fonksiyonu çalışır. Döndürdüğü float değeri önce bir temp değişkenine atılır. Bu değişken sqrt un içine koyulur ve sonuç bir double a yerleştirilir. Başka türlü sqrt fonksiyonu doğru sonucu veremezdi.

fclose(fptr);
printf("\nfile is now closed.\n");
}
printf("do you want to continue? y/n:");
scanf("%c", &selection);
scanf("%c", &selection);
}
printf("program ended. goodbye.");
return 0;
	İşlem tamamlandığı için file kapatılır.
	Kullanıcıya programı yeniden kullanma seçeneği verilir. Scanf in iki kere kullanılması gerekir çünkü printf in sonundaki \n karakterini yanlış algılamaktadır.
	Kullanıcı n girerse program kapatılır. Main sona erer.