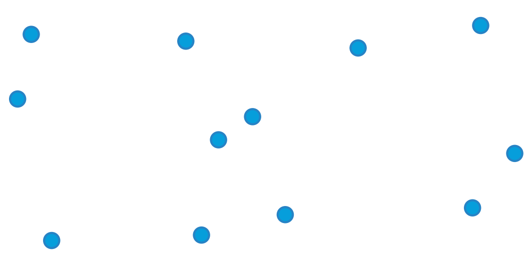
Problem najbližjih točk

Igor Pesek

V prispevku bomo predstavili problem najbližjih točk v množici P z $n \geq 2$ točkami. Rešitev problema je uporabna, ko moramo, denimo, odkriti oz. določiti, kateri dve osebi stojita najbližje ena drugi (otroška igra), pa tudi kateri dve letali sta si najbližje (preprečitev nesreče v letalskem prometu). Problem bomo opisali natančneje. Osebe bodo postale točke, igrišče pa bo postalo premica, ravnina ali prostor. Poiskati torej želimo najbližji točki v množici danih točk P, ki ležijo v N-dimenzionalnem prostoru. Najbližji v našem primeru pomeni običajno evklidsko razdaljo, kjer je razdalja med točkama v ravnini $p_1 = (x_1, y_1)$ in $p_2 = (x_2, y_2)$ enaka $dist_{p_1,p_2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Grafična ponazoritev problema je prikazana na sliki 1.



SLIKA 1.
Množica točk v ravnini

Pregled vseh možnih razdalj

Problema se lahko lotimo s preiskovanjem vseh možnih razdalj med točkami. V eni dimenziji, na premici, tako pregledamo $\binom{n}{2}$ parov točk in med njimi določimo najbližji par. Če podane točke naraščajoče uredimo, lahko preiskovanje pohitrimo, saj moramo pregledati le pare točk p_i in p_{i+1} za $i = 1, \ldots, n-1$ in med njimi določiti najbližjega. Takšno preiskovanje zahteva linearni čas, vendar moramo upoštevati še čas za urejanje množice točk. V ravnini in višjih dimenzijah nam ta razmislek ne pomaga, zato moramo izračunati razdalje za približno $O(n^2)$ parov točk.

Iskanje s pomočjo strategije deli in vladaj

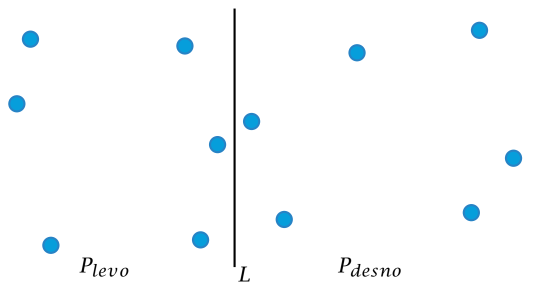
Problem lahko hitreje rešimo s pomočjo strategije deli in vladaj, ki smo jo v Preseku že predstavili. Pri tej strategiji se problem običajno rešuje s pomočjo rekurzivnih klicev v treh korakih; kar bomo predstavili za primer, ko točke ležijo v ravnini.

Deli. Poiščemo navpično premico L, ki deli množico točk na dva dela P_{levo} in P_{desno}. Ta sta po velikosti enaka oz. v primeru lihega števila točk v P je v P_{levo} ena točka več. Pri tem so vse točke iz P_{levo} levo od črte L in točke iz P_{desno} desno od črte L. Primer razdelitve s premico L je prikazan na sliki 2.

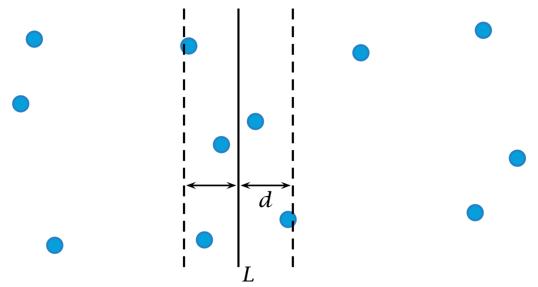
Vladaj. Izvedemo dva rekurzivna klica; s prvim poiščemo najbližji par točk v P_{levo} in z drugim klicem najbližji par točk v P_{desno}. Vsak klic vrne najkrajšo razdaljo d_{levo} oz. d_{desno} dane množice, kot je prikazano na sliki 3. V rekurzivnem klicu pazimo, da v primeru števila točk dane množice $|P| \leq 3$ za določitev najkrajše razdalje pregledamo vse možne razdalje. Sedaj določimo d, ki je enak

- $d = \min(d_{levo}, d_{desno})$.

Združi. Najbližji par točk je sedaj bodisi par z razdaljo d, ki smo jo določili z enim od rekurzivnih

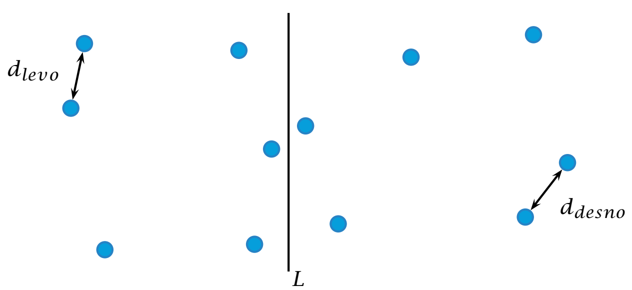


SLIKA 2.
Razdelitev na dve množici s premico L



SLIKA 4.
Navpični trak širine $2d$

klicev, bodisi par točk, kjer ena točka leži v P_{levo} in ena točka v P_{desno}. Le-teh z rekurzivnimi klici nismo zajeli. Vseh parov točk iz različnih množic nam ni potrebno preveriti. Zakaj ne? Iščemo namreč takšen par, katerega razdalja je krajša od d. Zadostuje torej da preverimo samo pare točk, kjer sta obe točki para znotraj navideznega navpičnega traku T, ki sega za razdaljo d na vsako stran od premice L. Skupaj je torej T širok $2d$, kot je prikazano na sliki 4. Zakaj je to dovolj? Če želimo najti krajšo razdaljo od d, je iskanje dlje od teh premic nesmiselno, ker bo razdalja zagotovo večja od d. V najslabšem primeru ena od točk leži na premici L, točka iz druge množice pa je največ za d oddaljena od prve točke. Torej nas vse točke, ki ležijo izven traku T, ne zanimajo in jih za ta korak v celoti pozabimo. Sledi pregled vseh parov točk, ki ležijo znotraj traku T. Ta korak lahko tudi pohitrimo. Kako? Točke uredimo naraščajoče



SLIKA 3.
Rezultat rekurzivnih klicev

po y-koordinati, kar nam pomaga pri preiskovanju, saj sedaj zadostuje, da pregledamo samo točke, ki tudi v navpični smeri niso oddaljene za več kot d od preiskovane točke. Primer je prikazan na sliki 5. Pregledamo vse točke znotraj traku T; če obstaja takšen $d_T < d$, potem smo našli novo najkrajšo razdaljo, sicer je najkrajša razdalja tista, ki smo jo dobili z enim od rekurzivnih klicev.

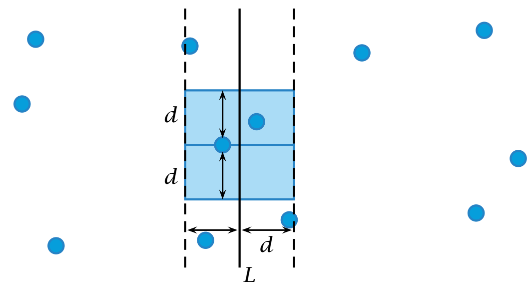
V nadaljevanju je predstavljen algoritem za iskanje najbližjega para točk. Nekatere funkcije v algoritmu so prepuščene bralcu, da jih napiše sam.

```
NajblizjiTocki (P, P_x ,n)
   if |P_x| <= 3: NajkrajsaRazdalja(P_x)
         //pregledamo vse pare

   Deli(P_x, P_levo, P_desno, L, n)
   d_levo = NajblizjiTocki(P, P_levo, n)
   d_desno = NajblizjiTocki(P, P_desno, n)
   d = min(d_levo, d_desno)

   //pregledamo vse točke znotraj traku T
   P_t = DoločiTočkeNaTraku(P_x, L, d)
   UrediPo_Y_Koordinati(P_t)
   d_t = PoisciNajkrajsoRazdaljo(P_t, d)

   return min(d, d_t)
```



SLIKA 5.
Področje iskanja v navpičnem traku T

Zaključek

Algoritem za iskanje najbližjega para po strategiji deli in vladaj ima časovno zahtevnost $O(n \cdot \log n)$, kar je precejšnja izboljšava v primerjavi z metodo preiskovanja vseh razdalj med pari točk, ki ima časovno zahtevnost $O(n^2)$. Predstavili smo delovanje algoritma v ravnini ($N = 2$), vendar se takoj postavi vprašanje, kaj pa v $3D$ prostoru ($N = 3$) ali kakšnem večdimenzionalnem prostoru ($N > 3$). Izkaže se, da algoritem deluje tako, da navpična premica ni več premica ampak ravnina oz. natančneje, delilna premica L je vedno prostor z eno dimenzijo manj, kot je N.

Literatura

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest in C. Stein, *Introduction to Algorithms*, MIT Press, 2009.

× × ×

Barvni sudoku

↓↓↓

→ V 8 × 8 kvadratkov moraš vpisati začetna naravna števila od 1 do 8 tako, da bo v vsaki vrstici, v vsakem stolpcu in v kvadratkih iste barve (pravokotnikih 2 × 4) nastopalo vseh 8 števil.

	3				7	4	8
	4			5			
						3	1
			5			8	4
5			4				
	1		7				
	7	4				2	
1			8				

→→→ REŠITEV BARVNI SUDOKU

2	3	5	6	1	7	4	8
7	4	8	1	5	3	6	2
4	8	7	2	6	5	3	1
3	6	1	5	7	2	8	4
5	2	6	4	3	8	1	7
8	1	3	7	2	4	5	6
6	7	4	3	8	1	2	5
1	5	2	8	4	6	7	3

× × ×