Proiectarea Algoritmilor

Ștefan Trăușan-Matu

stefan.trausan@cs.pub.ro

trausan@gmail.com

Obiectivele cursului

- Discutarea relaţiei dintre:
  - caracteristicile problemelor,
  - modul de rezolvare,
  - calitatea soluţiilor.

Obiectivele cursului

- Discutarea relaţiei dintre:
  - caracteristicile problemelor,
  - modul de rezolvare
  - calitatea soluţiilor =
    - cost minim (complexitate minimă)
    - corectitudine
    - aproximare bună

Obiectivele cursului

- Cunoașterea unui set important de algoritmi și metode de rezolvare a problemelor de algoritmică

Obiectivele cursului

- Prezentarea principalelor scheme de algoritmi,
- a performanțelor acestora și
- a problemelor la care se pot aplica

Obiectivele cursului

- Compararea variantelor unor algoritmi pentru rezolvarea problemelor dificile.

Obiectivele cursului

- Utilizarea teoriei predate la curs pentru
  - proiectarea algoritmilor de rezolvare pentru probleme
    - tipice și
    - dificile
  - întâlnite în practica dezvoltării sistemelor de programe.

Obiectivele cursului

- Dezvoltarea abilitatilor de adaptare a unui algoritm la o problema din viata reala

Obiectivele cursului

- Dezvoltarea abilitatilor de lucru in echipa

Obiectivele cursului

- Dezvoltarea CREATIVITĂȚII în proiectarea algoritmilor

Gândirea laterală http://www.edwdebono.com/lateral.htm

- **1. "You cannot dig a hole in a different place by digging the same hole deeper"**
- This means that trying harder in the same direction may not be as useful as changing direction. Effort in the same direction (approach) will not necessarily succeed.
- **2. "Lateral Thinking is for changing concepts and perceptions"**
- With logic you start out with certain ingredients just as in playing chess you start out with given pieces. But what are those pieces? In most real life situations the pieces are not given, we just assume they are there. We assume certain perceptions, certain concepts and certain boundaries. Lateral thinking is concerned not with playing with the existing pieces but with seeking to change those very pieces. Lateral thinking is concerned with the perception part of thinking. This is where we organise the external world into the pieces we can then 'process'.
- **3. "The brain as a self-organising information system forms asymmetric patterns. In such systems there is a mathematical need for moving across patterns. The tools and processes of lateral thinking are designed to achieve such 'lateral' movement. The tools are based on an understanding of self-organising information systems."**
- This is a technical definition which depends on an understanding of self-organising information systems.
- **4. "In any self-organising system there is a need to escape from a local optimum in order to move towards a more global optimum. The techniques of lateral thinking, such as provocation, are designed to help that change.“**

Gândirea paralelă http://www.edwdebono.com/lateral.htm

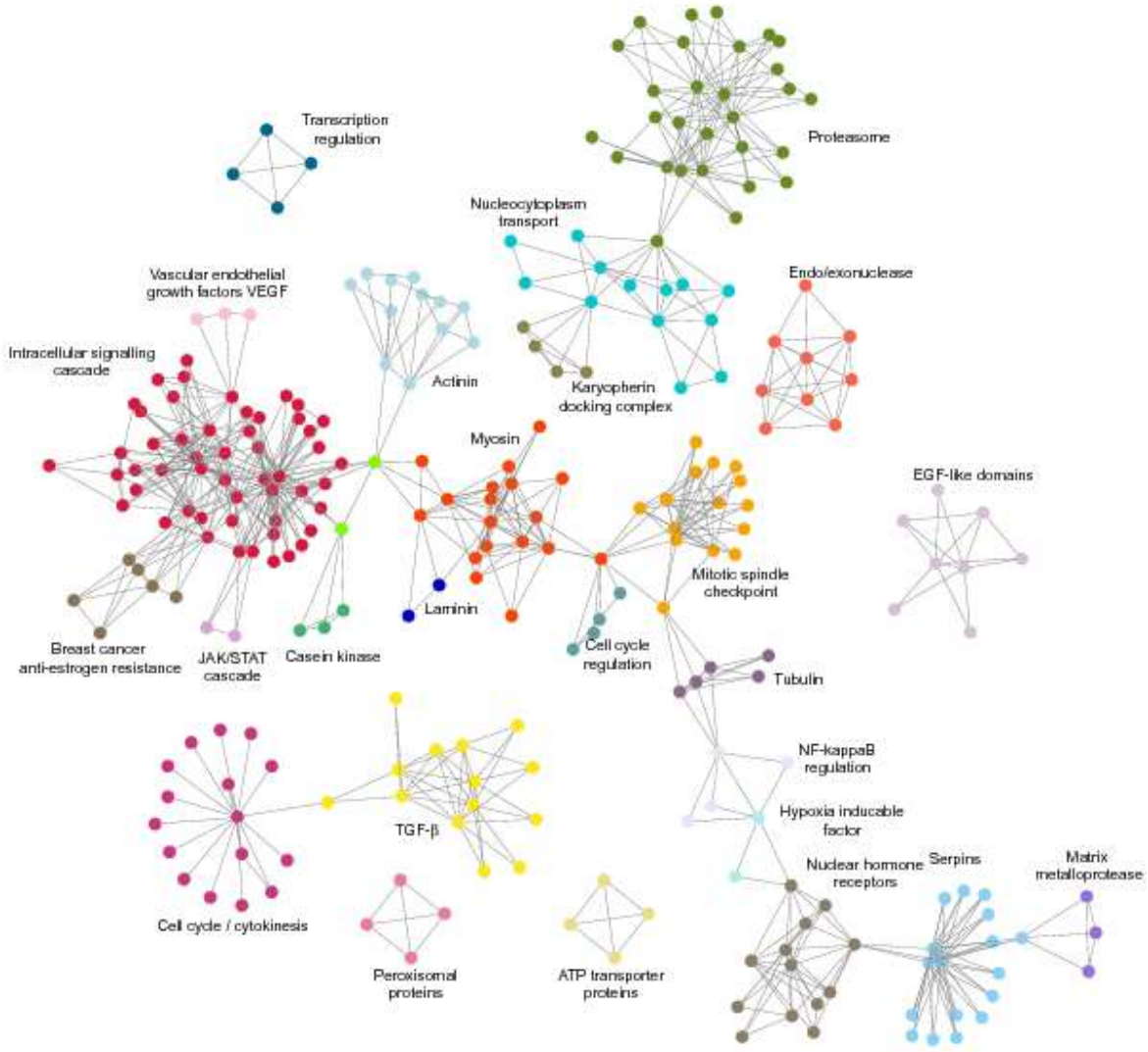
- **Parallel thinking is best understood in contrast to traditional argument or adversarial thinking.**
- With 'parallel thinking' both sides (or all parties0 are thinking in parallel in the same direction. There is co-operative and co-ordinated thinking. The direction itself can be changed in order to give a full scan of the situation. But at every moment each thinker is thinking in parallel with all the other thinkers. There does not have to be agreement. Statements or thoughts which are indeed contradictory are not argued out but laid down in parallel.In the final stage the way forward is 'designed' from the parallel thought that have been laid out.
- A simple and practical way of carrying out 'parallel thinking' is the Six Hats™ method which is now being used widely around the world both because it speeds up thinking and also because it is so much more constructive then traditional argument thinking.

Exemple de probleme

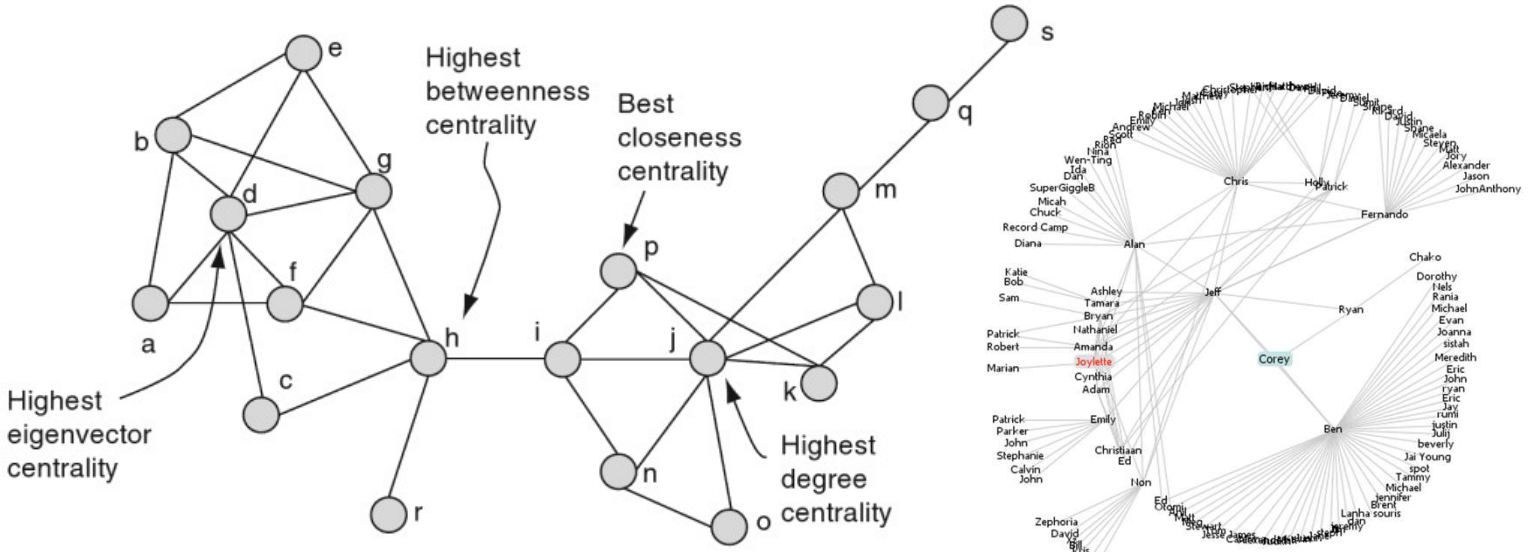
Probleme

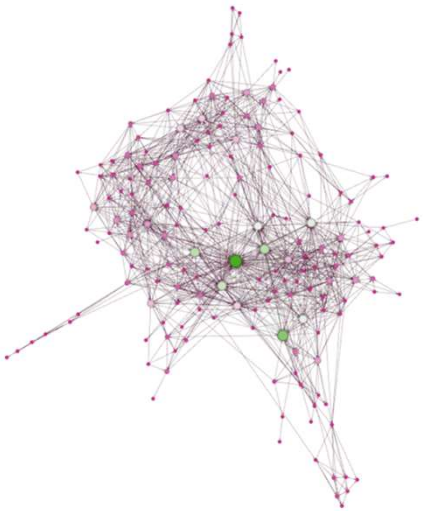
- Uzuale:
  - Alocare de resurse
  - Drumuri minime
  - Conectivitate
  - Optimalitatea rețelelor
  - Analiza erorilor

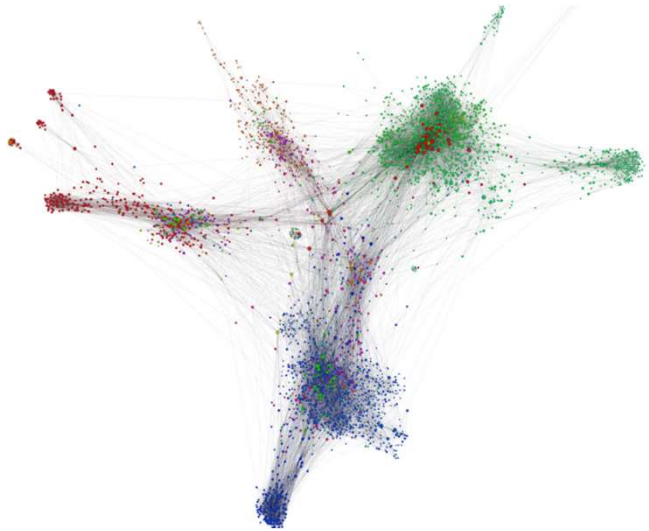
Rețele sociale



Analiza rețelelor sociale



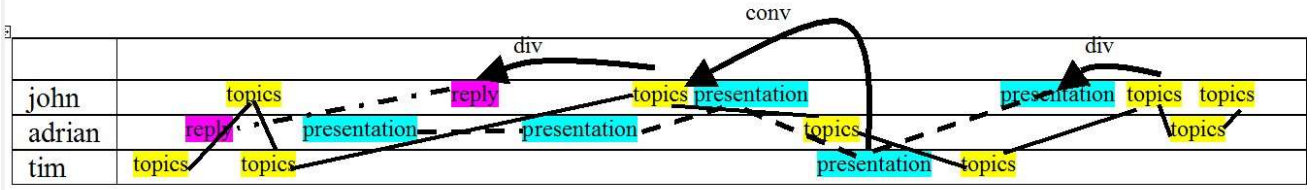




Analiza limbajului natural – Polyphonic analysis

(Trausan-Matu & Stahl, 2007, http://gerrystahl.net/vmtwiki/stefan.pdf)

Nr	Ref	Time	User	Text
17		10.26.25	tim	You discussed about a topic separation
18	15	10.26.37	adrian	First of all, the reply method is cumbersome
19	17	10.26.50	john	yes, because we did not like the way the topics were presented in concert chat
20	18	10.26.56	john	yes !!
21	20	10.27.04	john	i hate double-clicking!
22	20	10.27.18	tim	and how can we find topics ?
23	18	10.27.26	adrian	What bothers me is the linear presentation of the discussin
24	23	10.27.43	john	Yep
25	18	10.27.46	adrian	and double-clicking too
26		10.27.54	tim	You mean u want something like a chat forum ? :D
27	24	10.27.58	john	and the reply-to facility is supposed to help you
28	18	10.28.15	adrian	i'd like a tree presentation more
29	18	10.28.38	adrian	or maybe multiple chat columns, for each chat sub-thread
30	27	10.28.58	john	but it is really difficult to use in real-time, because there are so many topics discussed which intertwine each other
31	28	10.29.18	john	i subscribe to a tree-like presentation form
32	P 30	10.29.20	adrian	yes, that's why a clear separation of topics is needed
33	31	10.29.47	adrian	this is easy to implement, no problem here :)
34	30	10.29.49	tim	You need also a clever visual representation
35	30	10.30.05	tim	you'll need also a clever visual interface
36		10.30.22	tim	Who decides the topics ?
37	33	10.30.33	john	i suppose you are refering to the visual representation, right ?
38	37	10.30.45	john	What i would like is a clever way to separate the topics :)
39	38	10.30.59	john	not just doing ot myself, manually
40	37	10.31.00	adrian	Yeah
41	39	10.31.44	adrian	When you start a new thread (a new message, non-related to other message), the app can assume a new topic
42	39	10.31.46	john	i would like the application to be able to detect w topic change all by itself



Planul cursului (1)

- Scheme de algoritmi
  - divide&impera
  - rezolvare lacomă (Greedy) - arbori Hufmann
  - programare dinamică – AOC
  - backtracking cu optimizări
  - propagarea restricţiilor.

Planul cursului (2)

- Algoritmi pentru grafuri
  - parcurgeri,
  - sortare topologică,
  - componente tare conexe,
  - puncte de articulaţie, punţi,
  - arbori minimi de acoperire,
  - drumuri de cost minim,
  - fluxuri.

Planul cursului (3)

- Rezolvarea problemelor prin căutare euristică
  - A*
  - AO*
  - α-β
  - Completitudine şi optimalitate, caracteristici ale euristicilor.
- Algoritmi aleatorii
  - Las Vegas
  - Monte Carlo
  - aproximare probabilistică

Evaluare

- Examen 4 p
- Laborator 6 p

- 50% condiție de absolvire atât a laboratorului cât și a examenului

Bibliografie

- *Thomas H. Cormen, Charles E. Leiserson, Ronald R. Rivest.* Introducere in Algoritmi, Ed. Agora,
- *Cristian Giumale,* Introducere in Analiza Algoritmilor, Ed. Polirom 2004

ATENTIE!

- Prezentarile (“slide”-urile) de la curs sunt doar o parte din continutul cursului, nu sunt suficiente pentru pregatirea teoriei pentru examen – la curs se mai spun si lucruri in plus

CURS 1

Curs 1 - cuprins

- Scheme de algoritmi
- Divide et impera
  - Exemplificare folosind sortare prin interclasare (”merge sort”)

Scheme de algoritmi

Divide & Impera

Scheme de algoritmi

- Prin scheme de algoritmi intelegem tipare comune pe care le putem aplica in rezolvarea unor probleme similare
- O gama larga de probleme se poate rezolva folosind un numar relativ mic de scheme
- => Cunoasterea schemelor determina o rezolvare mai rapida si mai eficienta a problemelor

Divide et impera (1)

- Schemă generală de rezolvare de probleme (chiar și în viața cotidiană)

- Ideea (divide si cucereste) este atribuita lui Filip al II-lea, regele Macedoniei (382-336 i.e.n.), tatal lui Alexandru cel Mare si se refera la politica acestuia fata de statele grecesti

Divide et impera (2)

- Schema Divide et impera consta in 3 pasi la fiecare nivel al recurentei:
  - **Divide** problema data intr-un numar de subprobleme
  - **Impera (cucereste) –** subproblemele sunt rezolvate recursiv. Daca subproblemele sunt suficient de mici ca date de intrare se rezolva direct (iesirea din recurenta)
  - **Recombina –** solutiile subproblemelor sunt combinate pentru a obtine solutia problemei initiale

Divide et impera – Avantaje si Dezavantaje

- Avantaje
  - Produce algoritmi eficienti
  - Descompunerea problemei in subprobleme faciliteaza paralelizarea algoritmului in vederea executiei sale pe mai multe procesoare
- Dezavantaje
  - Se adauga un overhead datorat recursivitatii (retinerea pe stiva a apelurilor functiilor)

Exemplu - Merge sort (1)

- Algoritmul Merge Sort este un exemplu clasic de rezolvare cu ajutorul divide et impera
- **Divide:** Împarte secvența celor n elemente ce trebuie sortate in 2 secvente de lungime n/2
- **Impera:** Sorteaza secventele recursiv folosind *merge sort*
- **Recombina:** Secventele sortate sunt ansamblate pentru a obtine vectorul sortat
- Recurenta se opreste cand secventa ce trebuie sortata are lungimea 1 (un vector cu un singur element este intotdeauna sortat ☺)
- Operatia cheie este ansamblarea solutiilor partiale folosind interclasarea

Merge Sort (2)

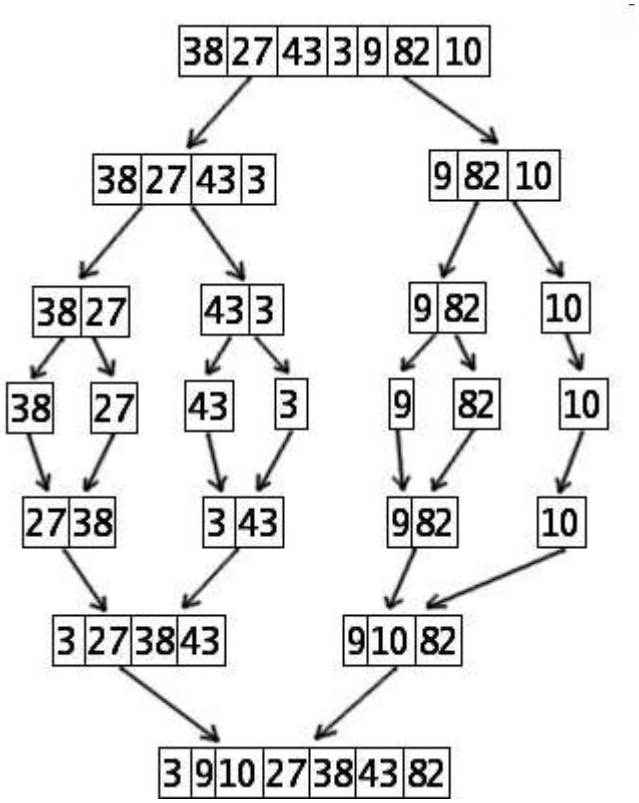
- Algoritm (adaptat din Cormen)
  - M-SORT(*A*, *p*, *r*)
  - **if** $p < r$
  - **then** $q \leftarrow [(p + r)/2]$ //divide
  - M-SORT(*A*, *p*, *q*) //impera
  - M-SORT(*A*, $q + 1$, *r*)
  - MERGE(A, p, q, r) //recombina

Merge Sort () – Algoritmul de interclasare

- Algoritm [Cormen]
  - MERGE(A, p, q, r)
  - 1 $n1 \leftarrow q - p + 1$
  - 2 $n2 \leftarrow r - q$
  - 3 create arrays $L[1 \text{ -> } n1 + 1]$ and $R[1 \text{ -> } n2 + 1]$
  - 4 **for** $i \leftarrow 1$ **to** $n1$
  - 5 **do** $L[i] \leftarrow A[p + i - 1]$
  - 6 **for** $j \leftarrow 1$ **to** $n2$
  - 7 **do** $R[j] \leftarrow A[q + j]$
  - 8 $L[n1 + 1] \leftarrow \infty$
  - 9 $R[n2 + 1] \leftarrow \infty$
  - 10 $i \leftarrow 1$
  - 11 $j \leftarrow 1$
  - 12 **for** $k \leftarrow p$ **to** r
  - 13 **do if** $L[i] \leq R[j]$
  - 14 **then** $A[k] \leftarrow L[i]$
  - 15 $i \leftarrow i + 1$
  - 16 **else** $A[k] \leftarrow R[j]$
  - 17 $j \leftarrow j + 1$

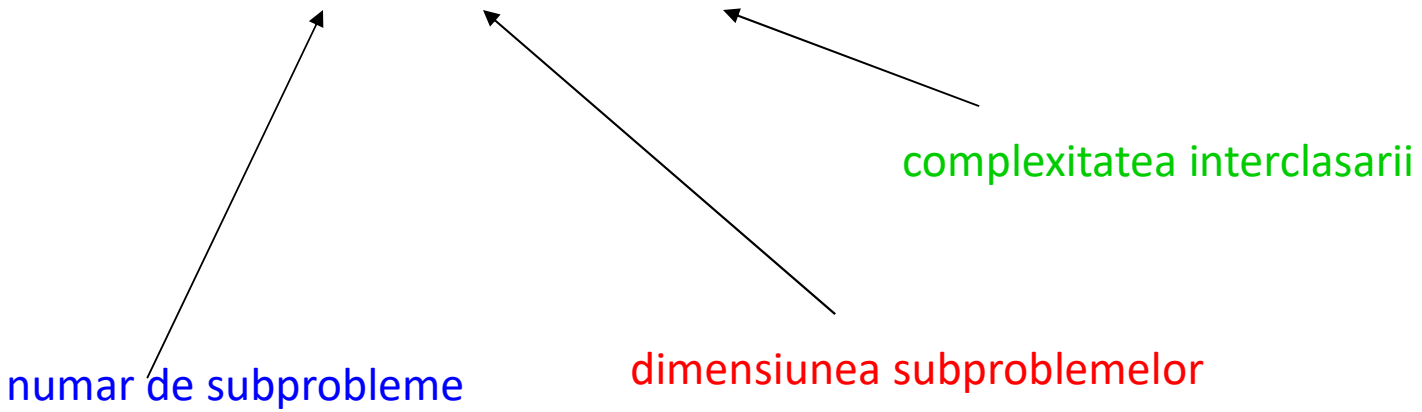
Exemplu functionare Merge Sort

- Exemplu functionare [Wikipedia]



MergeSort - Complexitate

- $T(n)=2T(n/2)+\Theta(n)$



=> (din T. Master) $T(n)=\Theta(n \log n)$

Divide et impera – alte exemple (I)

- Calculul puterii unui numar x^n
  - Algoritm “naiv”
    - pentru i=1->n rez=rez*x; return rez
      - complexitate Θ(n)

Discuție

Divide et impera – alte exemple (I)

- Calculul puterii unui numar x^n
  - Algoritm “naiv”
    - pentru i=1->n rez=rez*x; return rez
      - complexitate $\Theta(n)$
  - Algoritm divide et impera
    - daca n este par
      - return $x^{n/2}x^{n/2}$
    - daca n este impar
      - return $xx^{(n-1)/2}x^{(n-1)/2}$
    - complexitate: $T(n)=T(n/2)+\Theta(1)=>T(n)=\Theta(\log n)$

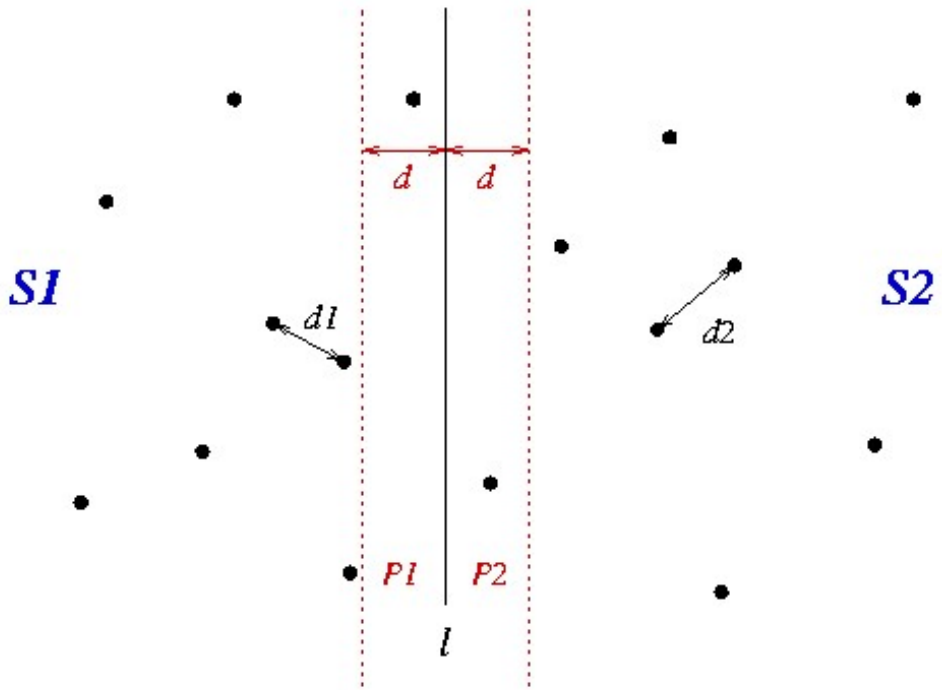
Divide et impera – alte exemple (II)

- Identificarea celei mai scurte distante intre 2 puncte din plan
  - algoritmul naiv – $\Theta(n^2)$

Discuție

Divide et impera – alte exemple (II)

- identificarea celei mai scurte distante intre 2 puncte din plan
  - algoritmul naiv – $\Theta(n^2)$



Divide et impera – alte exemple (III)

- sorteaza punctele in ordinea crescatoare a coordonatei x (O(nlog n))
- impartim setul de puncte in 2 seturi de dimensiune egala si calculam recursiv distanta minima in fiecare set (l= linia ce imparte cele 2 seturi, d = distanta minima calculata in cele 2 seturi)
- elimina punctele care sunt plasate la distanta de l >d
- sorteaza punctele ramase dupa coordonata y
- calculeaza distantele de la fiecare punct ramas la cei x vecini (nu pot fi mai multi)
- daca gaseste o distanta <d actualizeaza d

Discuție:

Când merge prost Divide&Impera?