Software engineering perspective on algorithmics

- main task - adaptibility -> complex compon. easy to change, modify
- algorithm component - nejsou určeny pro specificke pouziti, ale pro všechny kontexty teoreticky řešitelné daný alg. Alg A -> komponenta AC -> přepis (adaptace) každému možnému kontextu

1.1.1 Datové struktury - lower level

- teoreticky stanovit zakladni struktury pro algoritmy - list, set a podobne
- problemy:
  - nelze uspokojit vsechny potreby
  - otazka, co ma resit struktura a co algoritmus (mapovane uzly/hrany dvou grafu)
  - struktury musi byt genericke kvuli nezavislosti v konkretnich aplikacich - napr atributy uzlu a hran (cena, delka...) -> genericke atributy, problem:
    - stejne atributy pod ruznymy jmeny (kapacita, delka)
    - dva atributy pro stejny vyznam (delka, cas. doba)

1.1.2 Client code - higher level

- funkcionalita, implementace -> rozsiritelna a upravitelna
- teoreticky algoritmu - vzor pro konkretni algoritmy
  - vyber algoritmu pro subproblemy - meli by byt zamenitelne
  - vyber reprezentace vystupu - oznaceny strom / prosly strom hledani / poradi uzlu a pod
  - specialni techniky - pro vylepeni nebo zrychleni algoritmu
  - zabudovani nealgoritmicke funkcnoti

1.2 Test case - ilustracni priklad

1.2.1 pozadavky na flexibilitu

ruzne variace dat. struktury grafu

- abstrakce: neorintovany, bidirectional, orientovane atd..
- ruzne tridy grafu: planarni, biparitni, intervalove -> spec. struktura
- ruzne implementace v ruznych v ruznych kontextech
- implicitni reprezentace a hiearchicka struktura

1.2.2 ruznorodot numerickych typu

- pro delku hran, vzdalenot uzlu
- dve hodnoty, vice kriterii razeni, interval misto delky
- reprezentace nekonecne vzdalenoti, (int MAXVAL, nemozne hodnoty, boolean priznak)

1.2.3 variace prioritni fronty

- fibonaciho heap, d-heap, (stejne uzlu a hran), dial implementation

1.2.4 organizace uzlovych a hranovych atributu

- pripojeny record s atriuty k uzlu / hrane
- oddelene od grafove struktury
- v cache
- neni ulozeno, ale je vdy spocitano, napr podle druhu hrany

1.2.5 variace algoritmicke funkce

- koren - uzly
- koreny - uzly
- uzel - uzel

1.2.6 nealgoritmicke pozadavky

- format vtuou a vystupu
- animace, pocitani operaci, interaktvni ovladani behem behu
- robustnot - kontrola a overeni vstupu a vytupu

alg. **component:**

- adapting for new kontext straightforward task
- maintaince of sw with comp. easier
- casto pouzivane komponenty jsou spolehlive
- heuristicke metody - {neprectu} - moznost on/off

Jeden kompromis pro vsechny kontexty - v praxi kombinace aspektu - nelze pokryt verzemi pro vsechny oblati

1.3. General goals

1.3.1 lower level - underlying data structures

- not hard wired -> variable - conceptual polymorphism
- vlozena vrstva mezi komponentu a dat. strikturu - modifikace struktury se promitne jen do vrstvy, ne do cele komponenty
- vlozena vrstva z malych koncepcne polymorfnich komponent
    - slozena z toolboxu
    - vrstva muze tranformovat data ze struktur pro komponentu -> jedna komp pro vypocet (delka hran ze souradnic uzlu)

1.3.1 higher level - client code

- moznost klienta vyvolat context-pecific code behem behu alg
    - alespon pri kazde iteraci hlavni smycky
    - napr kuli interaktivni animaci a pod.

- stop points - klient ma moznost ukoncit beh alg nebo ho pausnout
- moznost cteni stavu ve stop points
    - napr vsechny polozky v priorini fronte s prioritami -> schopnost urcit jak se bude komponenta chovat dale
    - je potreba dodatecnych pristupovych metod
- pre a postprocesing jako customuzation - oddelit od hlavniho alg
- core loops nekolika spusteni algoritmu maji byt maji byt spojitelne do jedne smycky - simul. ochlazovani,
- vystup snadno upravitelny - zmena formatu vystupu s minnimalnim efektem na alg komponenty
- snadna zamena komponenty v kazdem kontextu z teoretickeho pohledu - napr pouziti strategy pattern

1.4. Exiting aproaches

1.4.1. lower level undergoing data structures

type specialization

- dedicnost a hiearchie
- problemy s ruznorodoti nekterych aspektu

component layers

- stredni vrstva sestava z vrstev propojenych komponent
- problemy se zamenou, pridanim aspektu, prilis "tucne"

linear iterators

- prochazeni linearcnich kolekci
- probleemy s grafy (hrany / uzly)
- accesing, validating

adjacency iterators

- iteroatory - po sousednich uzlech a hranach
- problemy: incidujici / sousedni uzel / hrana

aspect oriented programing

- jeden blok pro kazdy aspekt - samostatny
- komponenta slozena z vyberu apektu bloku

integrace algoritmu a datovych struktur

1.4.2 higheer level - client code

algoritmove komponenty jako typ s metodami pro ruzne kroky

strategy pattern

- snadne pro alg jako abstraktni datove typy
- zamena snadna, parametry v konstruktoru
- alg slozen z podalgoritmu

algorithmic generator and loop kernels

- alg poskytuje metodu pro jeden krok
- je treba smycky u klienta
- klient muze data vkladat do lib. truktury pro vystup

algorithms as iterators

- specialni pripad algorithmic generators, node iterator pro DFS / BFS

algorithm frameworks

- hlavni trida ma metody pro dany algoritmus
- metody predstavuji jednotlive zamenitelne casti
- odvozene tridy prepisuji metody pro specificky kontext
- core loops jsou ve frameworku - klient nema kontrolu nad behem alg

robustness oriented design

- komponenta zalozena na specifikaci
- overovani komponent vuci specifikaci

AOP - fussion aspects - spojeni vice mycek do jedne