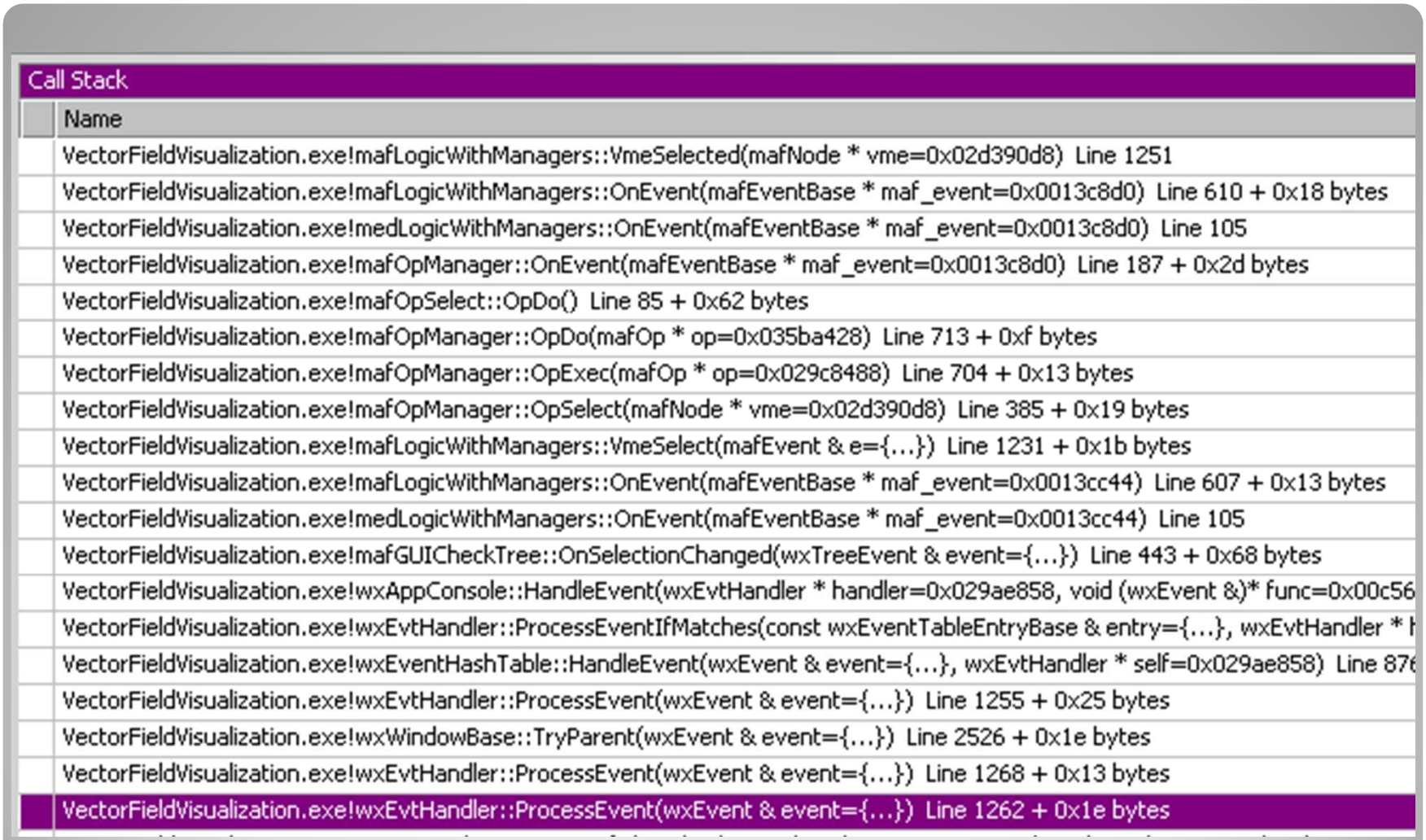
KIV/ZEP - 2011

Rekurze, dělení prostoru, medián

Rekurze

- přirozeným základem mnoha algoritmů
  - zejména těch využívajících D&C strategii
    - binární vyhledávání, quicksort, mergesort, semínkové plnění
  - Př: *Fibonacciho posloupnost*
- přímá vs. nepřímá rekurze
- pokud rekurze použita přímočaře
  - algoritmus často jednodušší
  - vyšší časové a pamětové nároky


Call Stack
Name
VectorFieldVisualization.exe!mafLogicWithManagers::VmeSelected(mafNode * vme=0x02d390d8) Line 1251
VectorFieldVisualization.exe!mafLogicWithManagers::OnEvent(mafEventBase * maf_event=0x0013c8d0) Line 610 + 0x18 bytes
VectorFieldVisualization.exe!medLogicWithManagers::OnEvent(mafEventBase * maf_event=0x0013c8d0) Line 105
VectorFieldVisualization.exe!mafOpManager::OnEvent(mafEventBase * maf_event=0x0013c8d0) Line 187 + 0x2d bytes
VectorFieldVisualization.exe!mafOpSelect::OpDo() Line 85 + 0x62 bytes
VectorFieldVisualization.exe!mafOpManager::OpDo(mafOp * op=0x035ba428) Line 713 + 0xf bytes
VectorFieldVisualization.exe!mafOpManager::OpExec(mafOp * op=0x029c8488) Line 704 + 0x13 bytes
VectorFieldVisualization.exe!mafOpManager::OpSelect(mafNode * vme=0x02d390d8) Line 385 + 0x19 bytes
VectorFieldVisualization.exe!mafLogicWithManagers::VmeSelect(mafEvent & e={...}) Line 1231 + 0x1b bytes
VectorFieldVisualization.exe!mafLogicWithManagers::OnEvent(mafEventBase * maf_event=0x0013cc44) Line 607 + 0x13 bytes
VectorFieldVisualization.exe!medLogicWithManagers::OnEvent(mafEventBase * maf_event=0x0013cc44) Line 105
VectorFieldVisualization.exe!mafGUICheckTree::OnSelectionChanged(wxTreeEvent & event={...}) Line 443 + 0x68 bytes
VectorFieldVisualization.exe!wxAppConsole::HandleEvent(wxEvtHandler * handler=0x029ae858, void (wxEvent &)* func=0x00c56
VectorFieldVisualization.exe!wxEvtHandler::ProcessEventIfMatches(const wxEventTableEntryBase & entry={...}, wxEvtHandler * h
VectorFieldVisualization.exe!wxEventHashTable::HandleEvent(wxEvent & event={...}, wxEvtHandler * self=0x029ae858) Line 87
VectorFieldVisualization.exe!wxEvtHandler::ProcessEvent(wxEvent & event={...}) Line 1255 + 0x25 bytes
VectorFieldVisualization.exe!wxWindowBase::TryParent(wxEvent & event={...}) Line 2526 + 0x1e bytes
VectorFieldVisualization.exe!wxEvtHandler::ProcessEvent(wxEvent & event={...}) Line 1268 + 0x13 bytes
VectorFieldVisualization.exe!wxEvtHandler::ProcessEvent(wxEvent & event={...}) Line 1262 + 0x1e bytes

Rekurze

Rekurze

- Odstraňování rekurze
  - vlastní uživatelský zásobník
    - odstraňuje problém přetečení zásobníku za cenu dalšího zpomalení kódu (alokace na haldě časově náročnější než alokace na zásobníku)
  - převod na smyčku
    - Př: *Fibonacciho posloupnost*
    - obtížné pro nepřímou rekurzi
      - možné řešení: popis dán jako řetězec symbolů
        - viz např. L-systémy
        - předmět KIV/KPG

- Binární hledání

```
bool bin_search(value, a[])
{
	if (a.length == 1)
		return a[0] == value;
	else
	{
		int middle = a.length / 2;
		//rozdel pole a na dve pole
		//a1 s prvky 0..middle - 1 a
		//a2 s prvky middle..a.length - 1
		if (a[middle] < value)
			return bin_search(value, a1);
		else
			return bin_search(value, a2);
	}
}
```

Rekurze

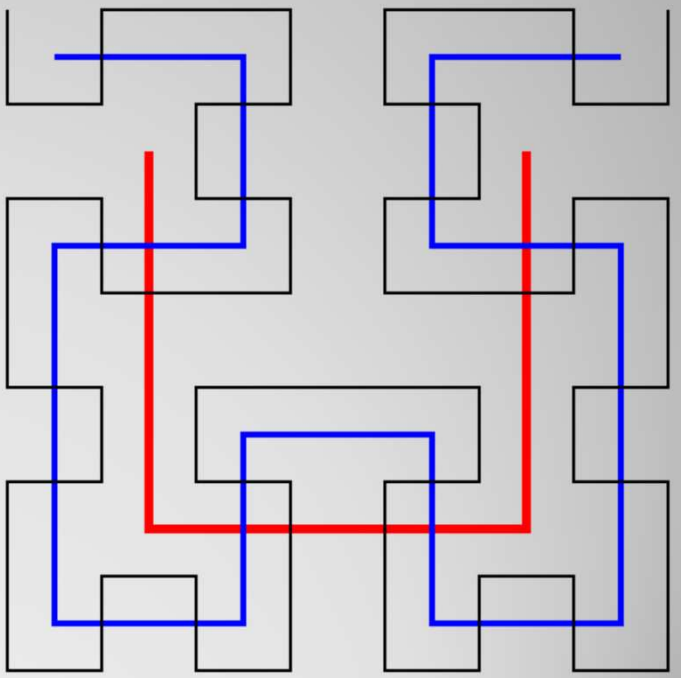
- Interpolační hledání
  - analogie hledání v tel. seznamu
  - může být rychlejší
    - pokud data jsou uniformně rozloženy (tj. při vynešení do grafu je trend lineární)

Rekurze

```
int interp_search(value, a[]){
  int low = 0;
  int high = a.length - 1;

  while (a[low] < value && a[high] >= value) {
   int mid = low + ((value - a[low]) *  (high - low)) / (a[high] - a[low]);

   if (a[mid] < value)
    low = mid + 1;
   else if (a[mid] > value)
    high = mid - 1;
   else
    return mid;
  }//end while

      return (a[low] == value) ? low : -1;
 }
```

Rekurze

- Hilbertova křivka
  - nejen pro okrasu
  - linearizace dat (2D na 1D)
  - podpora cache
    - položky nejsou seřazené, což může algortimům vadit a přesto blízko sebe
    - *Př: dělení prostoru pro body na triangulaci*



Rekurze

Rekurze

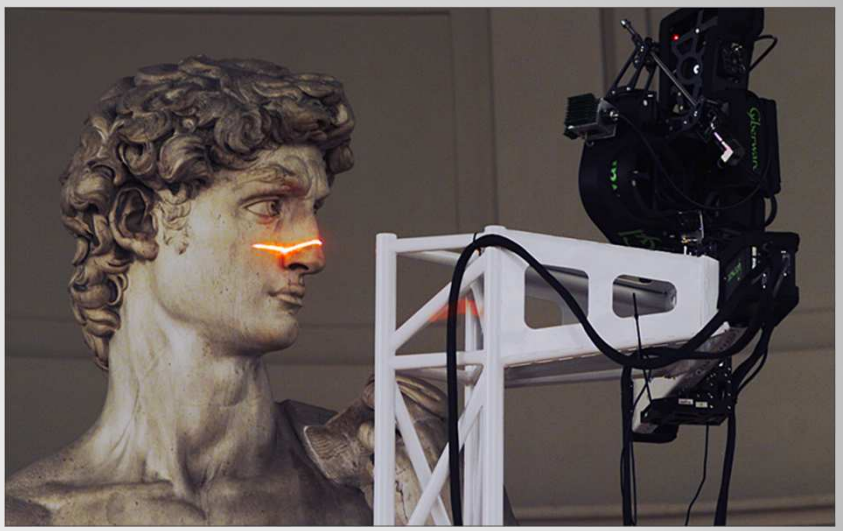
- Rozděl a panuj
  - je-li úloha jednoduchá vyřeš ji
  - jinak rozděl úlohu na dvě stejně složité podúlohy, vyřeš podúlohy (stejným způsobem) a stanov výsledek z dílčích výsledků
  - je-li dělení úlohy a slučování výsledků zvládnutelné v O(N) nebo lepším, pak máme složitost O(N*logN)
  - problém: jak dělit $\rightarrow$ často medián

Průměr, odchylka

- Průměr: $\bar{x} = \frac{1}{N}\sum x_i$
  - *Průměrné množství nápoje v hodpodě „U Černého vola“: 0.205, 0.202, 0.199, 0.198, 0.201, 0.195 = 0.2*
  - *Průměrný plat: 26, 30, 40, 16, 20, 16, 20, 80,*
- Odchylka: $\sigma = \sqrt{\frac{1}{N-1}\sum (x_i - \bar{x})^2}$
  - říká, jak moc blbý je průměr
    - hospoda: 0.003, plat: 21.38

Medián

- prostřední prvek (u sudého počtu prvků průměr dvou prostředních prvků)
  - *Platy: 16, 16, 20, 20, 26, 30, 40, 80*
- lépe se vypořádává s výkyvy
- odstraňuje tzv. outliers



Medián

- Výpočet aproximativní
  - Bucketing technika
    - nejprve min, max, pak v druhém průchodu zatřízení do bucketů -> histogram
  - Vzorkování
    - vyber reprezentativní vzorky a spočítej jejich přesný medián -> aproximace pro celek

Medián

- Výpočet přesný
  - brute-force: setřídit, vybrat → O(n·logn)
  - rozděl a panuj (D&C):
    - vyber pivota a roztřiď prvky do tří skupin tak, že skupina a1 obsahuje prvky menší než pivot, skupina a2 prvky rovny pivotu a skupina a3 prvky větší než pivot → T(n-1)
    - pokud medián leží ve skupině a1, zahoď a2 a a3 a rekurentně hledej v této skupině
    - analogicky pro případ, kdy medián leží v a3
    - leží-li v a2, jsme hotovi, medián = pivot

```
int mediansearch(median,a[])
{
    int x=a[median];
 // vytvoř tři pole a1[],a2[],a3[].
 // a1[] obsahuje prvky menší než x.
 // a2[] prvky rovny x.
 // a3[] prvky větší než x.

    if( a1.length >= median )
       return mediansearch(median,a1[]);
    if(a1.length + a2.length >= median )
       return x;
    else
       return mediansearch( median - (a1.length + a2.length),a3);
}
```

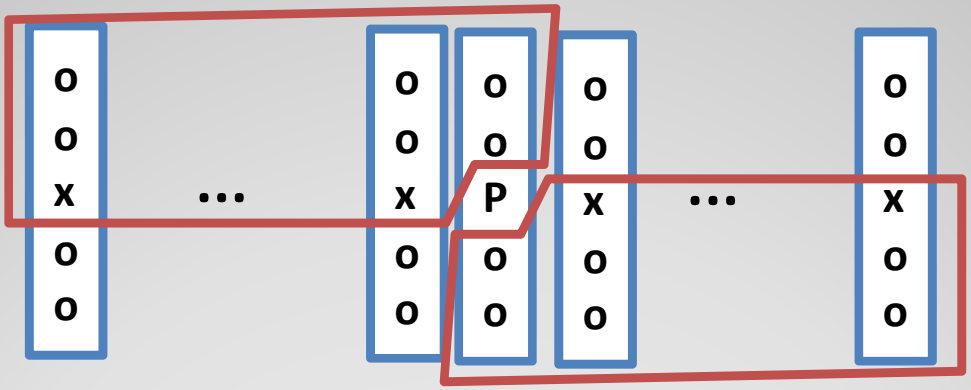
Medián

Medián

- Nejhorší případ: pivot vždy na kraji
  - $T(n) = T(n-1) + (n-1) \rightarrow O(n^2)$
- Nutno volit pivot tak, aby zajišťoval rozdělení v poměru 2:1 (1/3 zahazuji)
  - $T(n) = T(\frac{2}{3} \cdot n) + (n-1) \rightarrow$ zobecněný M.T.$\rightarrow O(n)$

Medián

1. rozděl posloupnost na pětice a nalezni mediány v každé pětici → c·n/5
2. nalezni medián těchto mediánů (přes D&C) → T(n/5), a tuto hodnotu použij jako pivot



- T(n)= c·n/5 + T(n/5) + T(7/10·n) → Θ(n)

Medián

- Proč zrovna pětice?
  - musí být lichý stupeň (jinak medián nepřesný)
  - zobecněný M.T: $T(n) = \sum T(a_i \cdot n) + f(n)$, $0 < a_i < 1$, x je výsledkem rovnice $\sum a_i^x = 1$ a $f(n) = \Theta(n^d)$, potom
    - $T(n) = \Theta(n^d)$, jestliže $x < d$
    - $T(n) = \Theta(n^x)$, jestliže $x > d$
    - $T(n) = \Theta(n^d \cdot \log n)$, jestliže $x = d$

	a1	a2	x
trojice	1/3	5/6	1.37
pětice	1/5	7/10	0.84
sedmice	1/7	5/7	0.76
devítice	1/9	13/18	0.71
jedenáctice	1/11	8/11	0.68

Geometrický medián

- Rozšíření pro množinu bodů v Euklidiovském prostoru
  - geom. medián je bod y takový, že

$$\operatorname*{argmin}_{y \in \mathbb{R}^n} \sum_{i=1}^{m} \|x_i - y\|$$

- Použití
  - všude tam, kde se pracuje s centroidem
- Výpočet
  - obtížný

- grafové algoritmy?

Příště