**11. Randomizované algoritmy typu Las Vegas a Monte-Carlo. Jejich složitost, srovnání s deterministickými. Algoritmus "Random Sample and Consensus" - RANSAC. Monte Carlo algoritmy pro výpočet vlastností prostorových objektů. Konstrukce bodů, přímek a rovin pomocí randomizovaných algoritmů.**

**Pravděpodobnostní (randomizované) algoritmy**

Pravděpodobnostní algoritmus dělá (na rozdíl od deterministického algoritmu) náhodné kroky, např. k některým krokům používá hodnoty získané z **generátoru náhodných čísel,** tím pádem dvě různá spuštění téhož pravděpodobnostního algoritmu na **stejných datech** mají (s velkou pravděpodobností) **různý průběh**.

Pravděpodobnostních algoritmů je mnoho typů, zde zmíníme jen dva a to algoritmy typu **Las Vegas** a typu **Monte Carlo** **.**

**Nedeterministický algoritmus** = algoritmus, který v každém svém kroku může volit z několika možností.

**Algoritmy typu Las Vegas**

Výsledek je **vždy správný**, náhodnost ovlivňuje **pouze dobu běhu** algoritmu, tj. po jaké cestě se algoritmus ke správnému výsledku dobere.

**Příklad**: randomizovaný QuickSort – od deterministické verze se liší náhodnými výběry **pivota** při každém dělení posloupnosti, což poskytuje následující výhody

* dává dobrý průměrný čas (tj. O(n log n)) i v případě, že data na vstupu nejsou náhodné permutace – žádný vstup není apriori špatný (pro každý deterministický výběr pivota existují apriori špatné vstupy)
* může být spuštěn **paralelně v několika kopiích**, výsledek je získán z kopie, kde výpočet skončí nejdříve (pro deterministickou verzi nemá takový postup žádný smysl)

**Algoritmy typu Monte Carlo**

Náhodnost ovlivňuje jak **dobu běhu**, tak **správnost výsledku**: algoritmus **může udělat chybu**, ale pouze jednostranně (u odpovědí ANO/NE) a s omezenou pravděpodobností. Např.: RANSAC.

**Výpočet hodnoty π**

Typickou úlohou metody Monte Carlo je zjištění hodnoty [Ludolfova čísla π](http://cs.wikipedia.org/wiki/P%C3%AD_%28%C4%8D%C3%ADslo%29). Základem je čtverec, kterému je vepsaná čtvrtkružnice. Analogicky jako u [Buffonovy jehly](http://cs.wikipedia.org/wiki/Buffonova_jehla) je možné jeho hodnotu zjistit náhodným házením jehlou, popřípadě jakýmkoliv jinými drobnými předměty do prostoru čtverce, a výsledný poměr počtu všech hodů a hodů do kruhu dá hodnotu čísla π.

|  |  |
| --- | --- |
| obsah čtvrtkružnice: S_1 = \frac{\pi r^2}{4}  obsah čtverce:  jejich poměr je pak :\frac{S_1}{S_2} =  \frac{\pi r^2}{4 r^2} = \frac{\pi}{4}  pak \pi = 4 \frac{S_1}{S_2} | *Čím víc bodů nastřílím, tím přesnější pi* |

**Příklad**: Rabin-Millerův algoritmus na testování prvočíselnosti

Úloha: pro zadané přirozené číslo n (rychle) rozhodnout zda je n prvočíslo

Trocha teorie (Malá) Fermatova věta (bez důkazu):

Nechť p je prvočíslo. Potom ∀ k ∈ {1,2, … ,p-1} platí kp-1 ≡ 1 (mod p)

Myšlenka: pokud n není prvočíslo, tak zkusíme (náhodně) najít „svědka“ k, porušujícího kn-1 ≡ 1 (mod n), který „dosvědčuje“, že n je opravdu číslo složené (není to prvočíslo)

Problém: pro některá složená čísla je svědků příliš málo, takže je příliš malá pravděpodobnost, že nějakého svědka (náhodně) vybereme.

Definice: Nechť T je množina dvojic přirozených čísel, kde (k, n) ∈ T pokud 0 < k < n a je splněna alespoň jedna z následujících dvou podmínek:

1. neplatí kn-1 ≡ 1 (mod n),
2. existuje i takové, že m = (n-1) / 2i je přirozené číslo a platí 1 < NSD(km-1-1, n) < n

Věta 1: Číslo n je složené tehdy a jen tehdy, když existuje k takové, že (k, n) ∈ T.

Věta 2: Nechť n je složené číslo. Pak existuje alespoň (n-1) / 2 takových čísel k, pro které platí (k, n) ∈ T.

**Rabin-Miller(n);**

for i:=1 to počet do

ki := náhodné přirozené číslo z intervalu [1,n-1];

if (ki,n) ∈ T then Report (n je složené);

Abort;

Report (n je prvočíslo)

Pokud Rabin-Miller(n) rozhodne, že n je **složené**, tak je to zaručeně správný výsledek (byl nalezen „svědek“), pokud Rabin-Miller(n) rozhodne, že n je **prvočíslo**, tak se **může jednat o chybu**, ale pouze v případě, že všechna vybraná ki byli „**ne-svědci**“ pro složené číslo n, což ale může (díky Větě 2) nastat nejvýše s pravděpodobností P(chyba) ≤ (1/2)počet pokud jsou výběry jednotlivých ki vzájemně nezávislé

Vlastnosti algoritmu:

* zvyšováním počtu iterací (počtu testovaných ki) lze dostat libovolně malou (předem zvolenou) pravděpodobnost chyby
* jednotlivé iterace (testy pro různá ki) lze provádět paralelně

Časová složitost:

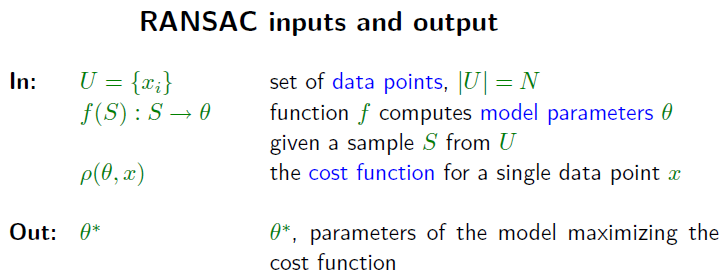
každá iterace trvá jen polynomiálně vzhledem k délce zápisu čísla n (tj. k délce vstupu), k tomu je ovšem potřeba ukázat, že test zda (ki,n) ∈ T je možno provést v čase polynomiálním v log n, což není triviální (je nutné mít další znalosti z teorie čísel)

**RANSAC (Random Sample and Consensus)**

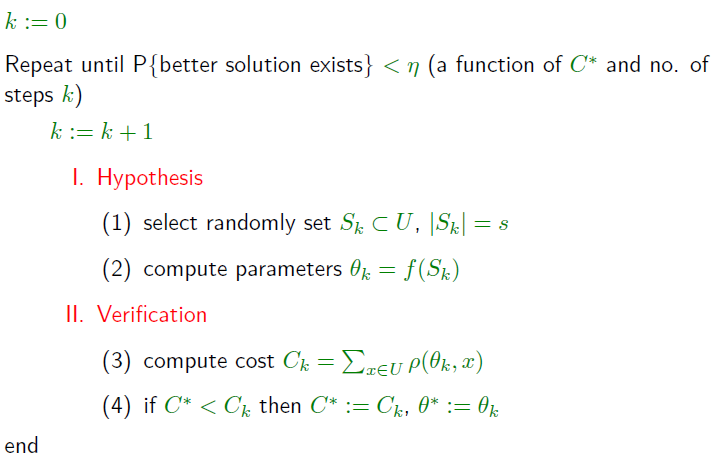
* published in 1981 as a model fitting method
* widely accepted as a method that works even for difficult computer vision problems

RANSAC is an abbreviation for "RANdom SAmple Consensus". It is an [iterative method](http://en.wikipedia.org/wiki/Iterative_method) to estimate parameters of a mathematical model from a set of observed data which contains [outliers](http://en.wikipedia.org/wiki/Outliers). It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, with this probability increasing as more iterations are allowed. The algorithm was first published by Fischler and Bolles in 1981.

A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be explained by some set of model parameters, and "outliers" which are data that do not fit the model. In addition to this, the data can be subject to noise. The outliers can come, e.g., from extreme values of the noise or from erroneous measurements or incorrect hypotheses about the interpretation of data. RANSAC also assumes that, given a (usually small) set of inliers, there exists a procedure which can estimate the parameters of a model that optimally explains or fits this data.



**RANSAC Algorithm**



**Explanation exmaple: line detection**

|  |  |  |
| --- | --- | --- |
| *Randomly select two points* | *The hypothesis model is the line passing trough the two points* | *The error function is a distance from the line* |
| *Points consistent wtih the model* | *Uncontaminated sample* | **Probability of selecting uncontaminated sample in *K* trials** |