Monte Carlo Tree Search

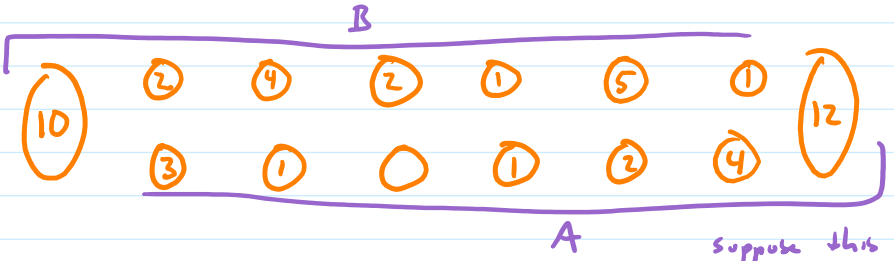
default policy: random

bias playout toward better moves

- move-averaged sampling technique (MAST)
  - maintain reward for actions during playout
- PAST (predicate averaged sampling techniques)

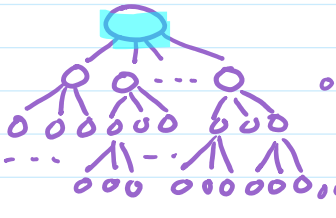
		n	r	
P_1:	A has more in store	~~100~~ 101	~~0.52~~	~~53/101~~
P_2:	B has more in store	50	0.4	
P_3:	A has no pit w/ >2 seeds	50	0.38	
P_4:	$\sim P_3$	~~100~~ 101	~~0.55~~	~~56/101~~
P_5:	A has empty pits	~~50~~ 51	~~0.7~~	~~36/51~~
P_6:	$\sim P_5$	100	0.3	

P_1 satisfied 100 · 0.52
P_4 satisfied +100 · 0.55
P_5 satisfied +50 · 0.70
250



suppose this is chosen in playout and leads to W

stochastic games:



tree policy: random at stochastic positions, UCB at other positions

imperfect information games:
determinization: pick values for unknown info

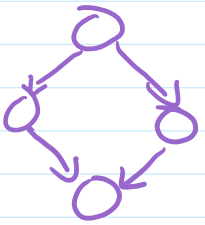
MCTS tree data → playability of game?

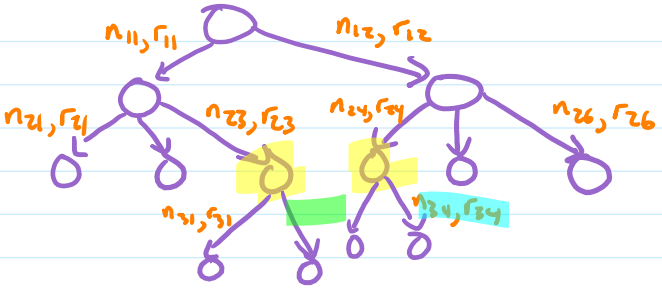
variance of score, spread

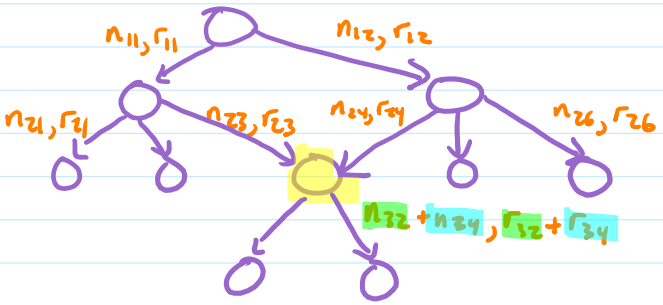
not too hard to win
not unbalanced to one player
not too long or too short

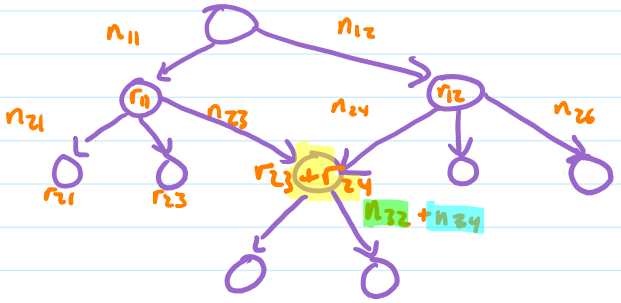
MCDS?

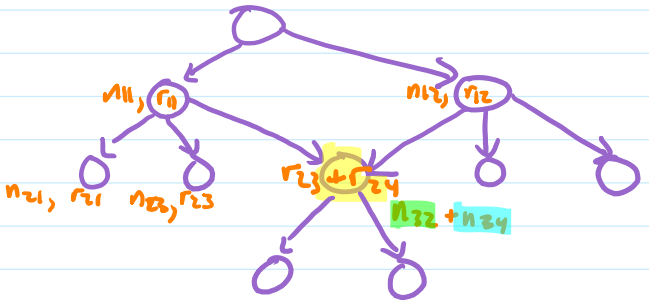
directed acyclic graph (DAG)











implementation: edges are implicit

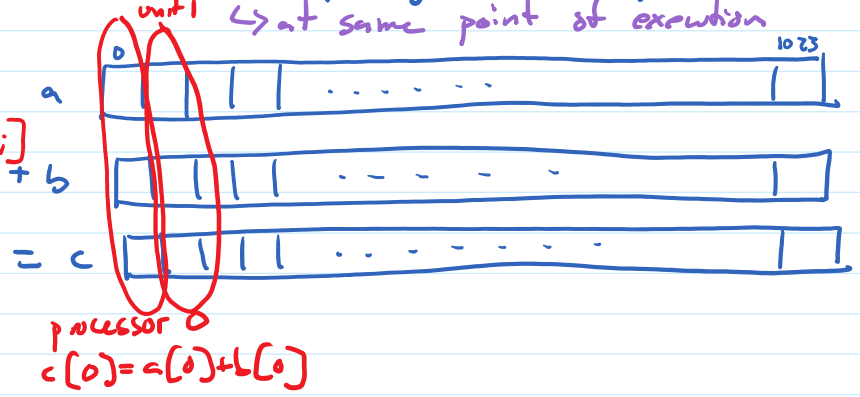
store as map pos → statistics

GPU Programming

task parallel : different tasks (functions) operating simultaneously on data
↳ each with independent point of execution

GPU → data parallel : same task operating simultaneously on different partition of data
↳ at same point of execution

```
for i=0 to n-1
    c[i] = a[i]+b[i]
```



matrix multiplication

$$\underset{n\times n}{\begin{pmatrix} a_{11} & a_{12} & \cdots \\ \vdots & & \\ a_{n1} & & a_{nn} \end{pmatrix}} \cdot \underset{n\times n}{\begin{pmatrix} b_{11} & b_{12} & \cdots \\ \vdots & \ddots & \\ & & b_{nn} \end{pmatrix}} = \underset{n\times n}{\begin{pmatrix} c_{11} & c_{12} & \cdots \\ \vdots & \ddots & c_{ij} \\ & & c_{nn} \end{pmatrix}}$$

processor (i,j) computes c_{ij} by

```
tot = 0
for k=1 to n
    tot += a_ik · b_kj
c_ij = tot
```