NC - $P(n) = n^{O(1)}$, $T(n) = \log^{O(1)}$ - polylog čas s polynomiálním počtem procesorů

PARTIME(T(n)) ≤ SPACE($T(n)^k$); SPACE(T(n)) ≤ PARTIME($T(n)^{k'}$); k a k' konstanty; PARTIME(f(n)) paralelně v čase O(f(n)), SPACE(f(n)) sekvenčně v prostoru O(f(n))

Algoritmus		čas	# proc	model	šířka slova	
⌊log n⌋	$v := \lfloor \frac{n}{2^i} \rfloor$; if (v > 0) and ($\lfloor \frac{v}{2} \rfloor$=0) then $\bar{r}_1 := i$	$O(1)$	⌊log n⌋	CREW	$O(\log n)$	
⌈log n⌉	$v := \lfloor \frac{n}{2^i} \rfloor$; if (v > 0) and ($\lfloor \frac{v}{2} \rfloor$=0) then if n = 2^i then $\bar{r}_1 := i$ else $\bar{r}_1 := i+1$	$O(1)$	⌊log n⌋	CREW	$O(\log n)$	

Max v konst čase	PID → (log n)-bitová čísla *i* (tým) a *j* (člen)
	if *j = 0* then $\bar{r}_{n+1+i} := 0$
	if ($x_i < x_j$) then $\bar{r}_{n+1+i} := 1$
	if (*j = 0*) and ($\bar{r}_{n+1+i} = 0$) then $\bar{r}_1 := \bar{r}_{i+1}$
		$O(1)$	n^2	COMMON	min.
					2 log *n*
						Búno: n je mocnina 2

Sčítání *n* čísel s *b* bity $x_0,...,x_{n-1}$	součet *n* čísel s b bity nebude míť víc než O(*b* + log *n*) bitů: log n viz alg. vyššie b spočítáme max. a použijeme horní celou část logaritmu $2^{n(b+\log n)}$ týmů po *n* proc.; PID → $y_0, \ldots, y_{n-1}$ s *b* + log *n* bity a *j* s log *n* bity → 0-tý člen týmu kontroluje, jestli $y_0 = x_0$ → j-tý člen týmu kontroluje, jestli $y_j = y_{j-1} + x_j$ právě jeden tým nezjistí žádnou nerovnosť – jeho manažer extrahuje y_{n-1} - výsledek	$O(1)$	$n2^{n(b+\log n)}$	COMMON	n(b+log n)+ log n	Búno: n je mocnina 2
Násobení dvou *b* bitových čísel	zjisti *b* (log většího z nich) → $\bar{r}_0$ for i in parallel do extrahuj *i*-tý bit čísla x_2 (odpovídá řádu 2^i) když je bit 1, posuň x_1 o *i* bitů doleva (tj. násob 2^i) → $\bar{r}_{i+1}$ zapiš 0 jinak do $\bar{r}_{i+1}$ zapiš 0 součet $\bar{r}_1, \ldots, \bar{r}_b$ spočítáme v konst. čase	$O(1)$	$b2^{b(2b+\log b)}$	COMMON	O(b(b+log b)) = $O(b^2)$	
Odmocňování	*n* týmů tým *i* kontroluje, zda *i* = $\lceil n^{\frac{1}{c}} \rceil$ počítá i^c pomocí *c-1* násobení; jeden tým má $2^{O(\log^2 n)}$ procesorů if ($n \le i^c$) and ($i^{c-1} < n$) then $\bar{r}_1$:= *i*	$O(1)$	$n2^{O(\log^2 n)}$	COMMON	$O(\log^2 n)$	c>1 ∈ N n ∈ Z → $\lceil n^{\frac{1}{c}} \rceil$

Sčítání s menší šířkou slova	Vstup: $n \in$ N s $n^{1-\frac{1}{c}}$ bity pro nějaké celé $c \geq 2$ součet a každý částečný součet má max. $O(n^{1-\frac{1}{c}})$ bitů spočítej $m = n^{\frac{1}{c+1}}$ 1. rozděl vstupní čísla do skupin po m číslech a sečti každou skupinu; opaku c-krát → dostaneme $\frac{n}{m^c}$ (šířka slova: $m \cdot n^{1-\frac{1}{c}} = n^{1-\frac{1}{c^2+c}}$) 2. sečti $\frac{n}{m^c}$ částečných součtů (šířka slova: $\frac{n}{m^c} \cdot n^{1-\frac{1}{c}} = n^{1-\frac{1}{c^2+c}}$)	$O(1)$		COMMON	$O(n^{1-\frac{1}{c^2+c}})$	
prev	prev(x_1, … ,x_n) = <y_1, … , y_n> kde y_i = ● j pokud $x_j = x_i$, $j < i$ a x_k != x_i pro $j < k < i$ ● 0 pokud takové j neexistuje n^2 týmů - tým odpovídá dvojici <i, j>, $1 \leq i, j \leq n$ 1. komunikační registr pro každý tým, inicializován na 0 2. k-tý procesor, $j < k < i$ (ostatní nepracují) ověří, zda x_k != x_i, pokud zjistí rovnost, zapíše 1 do kom. registru 3. manažer kontroluje kom. registr týmu, pokud je tam 0 a $x_j = x_i \rightarrow \bar{r}_1$:= j	$O(1)$	n^3	COMMON	$O(log\ n)$	
last	last(x_1, … ,x_n) = <y_1, … , y_n> kde y_i = ● j pokud $x_j = x_i$ a x_k != x_i pro $j < k \leq n$ ● 0 pokud takové j neexistuje	$O(1)$	n^3	COMMON	$O(log\ n)$	
Tranzitivní uzávěr grafu	Vstup: orientovaný graf G = (V, E), V = {1, . . . , n}, E je mn. orientovaných hran zadaná maticí sousednosti - A[i, j] = true ⇔ (i, j) ∈ E. Výstup: graf tranzitívneho uzáveru G' = (V, E'), kde (x, y) ∈ E' ⇔ v G existuje cesta z x do y. 1. PID → | i | j | k | a. i := $\lfloor \frac{PID}{n^2} \rfloor$ b. j := $\lfloor \frac{PID \bmod n^2}{n} \rfloor$ c. k := $PID \bmod n$ 2. A[i, i] := true 3. log n times do if A[i, k] and A[k, j] then A[i, j] := true	$O(log\ n)$	n^3	COMMON	$O(log\ n)$	

Min - vyvážený bin. strom	for k := m − 1 downto 0 do
	for all j, $2^k \leq j < 2^{k+1}$ in parallel do
	A[j] := min(A[2j], A[2j + 1])
		$O(log\ n)$	$\frac{n}{2}$	EREW		

Technika zdvojování	for all k ∈ L in parallel do
	P(k) := next(k);
	If P(k) != k then distance(k) := 1
	else distance(k) := 0
	Repeat log n times
	For all k ∈ L in parallel do
	If P(k) != P(P(k)) then
	distance(k) := distance(k) + distance(P(k));
	P(k):= P(P(k))
	For all k ∈ L in parallel do rank(k) := distance(k);
		$O(\log n)$	n	CREW		distance → value,
						+ → nějaká binární, asociativní operace ⊕ (např. max, min…)

Prefixové výpočty	*n* čísel uložených v poli *A* – A[n], A[n + 1], . . . , A[2n − 1]
	prefixový výpocet spočítá hodnoty A[n], A[n] ⊕ A[n + 1], A[n] ⊕ A[n + 1] ⊕ A[n + 2
	. . . , kde ⊕ je asociativní binární operace
	for k := m − 1 downto 0 do
	for all j, $2^k \le j \le 2^{k+1} - 1$ in parallel do A[j] := A[2j] ⊕ A[2j + 1];
	B[1] := A[1];
	for k := 1 to m do
	for all j, $2^k \le j \le 2^{k+1} - 1$ in parallel do
	if odd(j) then B[j] := B[$\frac{j-1}{2}$]
	else if j = 2^k then B[j] := A[j]
	else B[j] := B[$\frac{j}{2}$ − 1] ⊕ A[j];
						použije se metoda binárního stromu,
						nechť n = 2^m, pro vhodné celé m > 0

Minima intervalů	Vstup: vektor celých čísel x(1), …, x(n) a vektor intervalů int(i) = (l(i)..r(i)), oba velikosti n Výstup: pro každé i spočítat min{x(k); k ∈ int(i)}. 1. v každém uzle spočítáme minimum z jeho potomků – dostaneme minima v “dobrých” intervalech 2. každému intervalu int(i) = (l(i)..r(i)), se přidělí 1 procesor, ten udělá rozklad na “dobré” intervaly a spočíta z nich minimum					metoda binárního stromu; n mocnina dvojky, x(i) uložené v listech
Euler cyklus na stromě	1. každý seznam sousedů zacyklíme zdvojováním 2. for all (i, j) ∈ E in parallel do tournext(i, j) := next(j, i)	$O(\log n)$	n			
Zakořenění stromu, DFS	Eul. cyklus přetrhneme v nějakém vrcholu → seznam prohledávání do hloubky očísluj hrany v seznamu od 1: rank(r, s) je pořadové číslo hrany (r, s) v Eul. cestě neorient. hrana {r, s} se vyskytuje jako (r, s) a (s, r): s menším číslem postupová, druhá návratová pro každou postupovou hranu (i, j) dej father(j) := i					

Počet potomků vrcholu v včetně	for all i in parallel do nd(i) := $\frac{rank(i, father(i)) - rank(father(i), i) + 1}{2}$	$O(\log n)$	n			
Preorder číslování	spočítej podseznam postupových hran očísluj vrcholy: preorder(j) := číslo hrany (father(j), j) + 1 počáteční vrchol Eul. cesty bude mít číslo 1 důsledek: i je předkem j ⇔ preorder(i) ≤ preorder(j) < preorder(i) + nd(i)					

Délka nejkratší cesty v grafu	Vstup: G = (V, E), M[i, j] ≥ 0 matice ohodnocení hran ∀i, j ∈ V
	Výstup: M'[i, j] = 0 pro i = j; M'[i, j] = min{M[i0, i_1] + M[i_1, i_2] + · · · + M[i_{k-1}, i_k]}
	minimum se počítá přes všechny posloupnosti i_0, i_1, …, i_k, kde i_0 = i a i_k = j
	
	for all i, j in parallel do m[i, j] := M[i, j]
	repeat log n times
	for all i, j, k in parallel do
	q[i, j, k] := m[i, j] + m[j, k];
	for all i, j in parallel do
	m[i, j] := min{m[i, j], q[i, 1, j], q[i, 2, j], …, q[i, n, j]};
	for all i, j in parallel do
	if i != j then M'[i, j] := m[i, j]
	else M'[i, j] := 0
		$O(\log^2 n)$
		CREW
		
		$O(\log n)$
		COMMON
			$O(\frac{n^3}{\log n})$
			CREW
			
			$O(n^4)$
			COMMON
				CREW
				
				COMMON		

Komponenty souvislosti grafu	Vstup: Matice sousednisti neorient. grafu G = (V, E), kde V = {1, …, n} Výstup: vektor C: C[i] = C[j] = k ⇔ i a j patří do stejné komponenty a k je nejmenší číslo vrcholu z této komponenty for all i ∈ V in parallel do C[i] := i repeat log n times Krok I - Vytváření stromových cyklů for all i ∈ V in parallel do T[i] := $\min_{j \in V}$ {C[j] | (A[i, j] = 1) & (C[j] != C[i]) } for all i ∈ V in parallel do T[i] := $\min_{j \in V}$ {T[j] | (C[j] = i) & (T[j] != i) } Krok II - Stahování stromových cyklů do zakořeněných hvězd for all i ∈ V in parallel do B[i] := T[i] repeat log n times for all i ∈ V in parallel do T[i] := T[T[i]] for all i ∈ V in parallel do C[i] := min{ B[T[i]], T[i] }	$O(\log^2 n)$	$O(\frac{n^2}{\log n})$ lze s $O(\frac{n^2}{\log^2 n})$ po každé iteraci vnějšího cyklu komprim. graf na aktivní p-vrcholy, které nepokrývají celé komp.			vrcholy i s rovnakou hodnotou C[i] tvoří tzv. pseudovrchol i reprez. p-vrcholem C[i]

Vylepšený alg. komponent souvislosti	Vstup: matice sousednosti A velikosti n × n
	Výstup: pole D velikosti n, D[i] =nejmenší číslo vrcholu v komp. souvislosti, ve které leží vrchol i
	
	A_0 := A; n_0 := n; k := 0;
	while n_k > 0 do
	k := k + 1;
	1. C[i] := min { j | (A_{k-1}[j, i] = 1) & (i != j) }, pokud takové j neexistuje, ta i
	2. stáhni pseudovrcholy def. prostřednictvím C na zakořeněné hvězdy
	3. každý kořen netriviální hvězdy označ jako "nový p-vrchol" a očísluj je; r(i) označuje číslo vrcholu i
	4. n_k :=počet nových p-vrcholů
	5. vytvoř matici A_k tvaru n_k × n_k - matici sousednosti pro nové p-vrcholy
	6. pro každý vrchol urči D[i] – rovná se i, když C[i] = i, jinak opačným postupem než (5) expanduj každý p-vrchol v na pseudovrchol a pro všechny prvky j pseudovrcholu D[j] := v
		$O(log^2 n)$	$O(\frac{n^2}{log^2 n})$			

Minimální kostra grafu	Pro každý vrchol u ∈ V nechť {u, C(u)} je nejlevnější hrana z u. Potom existuje minimální kostra grafu G, která obsahuje (najednou) všechny hrany {u, C(u)} pro všechny vrcholy u ∈ V. C definuje les stromových cyklů délky 2. algoritmus analogický algoritmu pro komponenty souvislosti v kroku (1) C[i] := j takové, že cena(i, j) = min { cena(i,k) | i != k } uchování reprezentanta v kroku (5): 1. fáze: pro každý vrchol uspořádat hrany do "nových p-vrcholů" podle čísla p-vrcholu potom pro každý vrchol v každé skupině se stejným p-vrcholem vybrat minimum podle ceny ⇒ z matice rozmerů n_{k-1} × n_{k-1} dostaneme matici sousednosti rozmerů n_{k-1} × n_k – hrany s minimální cenou z každého vrcholu do nových p-vrcholů čas $O(log\ n_{k-1})$ s $O(n_{k-1}\frac{n_{k-1}}{log\ n_{k-1}})$ procesory, z toho třídění na začátku se dá udělat v čase $O(log\ n_{k-1})$ s $O(n_{k-1})$ procesory; počítání minim – prefixový výpočet 2. fáze: transponovaně stejný postup ⇒ z matice rozmerů n_{k-1} × n_k dostaneme matici sousednosti rozmerů n_k × n_k – hrany s minimální cenou z každého nového p-vrcholu do nových p-vrcholů	$O(log^2 n)$	$O(\frac{n^2}{log^2 n})$			

2-souvislé komponenty	Vstup: G = (V, E) neorientovaný graf zadaný maticí sousednosti Výstup: pole B[e] = B[f], pro hrany e, f ∈ E, ⇔ e, f leží ve stejné 2-souvislé komponentě grafu G Pomocný graf: G' = (E, $\overline{E}$) – vrcholy sú hrany grafu G, T = (V, E') je kostra, p(v) je otec vrcholu v; do $\overline{E}$ dáme hrany tvaru i. {{u, p(u)}, {u, w}}, ak {u, w} ∈ E \ T, u < w a u není kořen ii. {{u, p(u)}, {w, p(w)}}, ak {u, w} ∈ E \ T, u a w jsou neporovnatelné (jeden není následníkem druhého) iii. {{w, p(w)}, {p(w), p(p(w))}}, pokud ani w, ani p(w) není kořen a existuje hrana {z, y} ∈ E \ T taková, že z je následníkem w a y není následníkem p(w) low(v) = nejmenší číslo vrcholu, do kterého vede hrana z nějakého potomka vrcholu v high(v) = největší číslo vrcholu, do kterého vede hrana z nejakého potomka vrcho v podmienka (iii) ≡ (low(w) < p(w)) & (high(w) ≥ preorder(p(w)) + nd(p(w))) low(v) a high(v) pro všechny vrcholy se dá spočítat metodou minim na intervalech dokonce na EREW PRAM v čase $O(\log n)$ s $O(n)$ procesory G'' je podgraf G' obsahující jen hrany typu (ii) a (iii) G'' má maximálně n − 1 uzlů (hrany z T) G'' má méně než $(n-1)^2$ hran najdeme komponenty souvislosti grafu G'' pro každou nekostrovou hranu najdeme nějakou kostrovou hranu, ke které je připojená (hrany typu (i)) a přidáme ji do její komponenty souvislosti	komponenty souvislosti grafu (V, E) v čase f(|V|, |E|) na modelu M 2-súvislé komponenty grafu (V, E) v čase O(f(|V|, |E|) + log |V|)	komponenty souvislosti grafu (V, E) s g(|V|, |E|) procesory na modelu M 2-súvislé komponenty grafu (V, E) s O(g(|V|, |E|) + |E| + |V|) procesory	(M ∈ {EREW, CREW, CRCW}		hrany e, f patří do stejné komponenty 2-souvislosti neorient. grafu ≡$_{df}$ existuje prostý cyklus obsahující e, f

Zorientování Euler. grafu	Vstup: G = (V, E) neorientovaný alebo orientovaný Eulerovský graf
	Výstup: Euler. cyklus procházející všemi hranami
	Předzpracování pre neorientovaný graf H = (V, E) – volba orientace hran: hranu {i, j} nahradíme dvojicí orientovaných hran (i, j) a (j, i); jednu z nich musíme "zahodit'"
	● reprezentace: seznam hran EDGE v poli velikosti 2|E|
	● EDGE lexikograficky utřídíme – čas $O(log \|E\|)$ s $O(\|E\|)$ procesory dostaneme EDGE', kde hrany vycházející z jednoho vrcholu v jdou za sebou a můžeme je oindexovat (v, v0), (v, v1), . . . , (v, vd−1), kde d je stupeň vrcholu v v grafu H
	● definujeme následníka hrany: pro každé v ∈ V a i liché
	SUCCESSOR[(v_i, v)] := (v, v_{i+1})
	SUCCESSOR[(v_{i+1}, v)] := (v, v_i) (i + 1 se počítá modulo d)
	tím se vytvoří cykly s tím, že pokud tam bude cyklus C, tak tam bude i opačně orientovaný cyklus C' přes stejné vrcholy
	● právě jeden z dvojice cyklů C, C' zrušíme: např. lexikograficky utřídíme hrany cyklu a ten z cyklů C, C', který má lexikograficky větší minimální hranu zrušíme a jeho seznam hran vyradíme z EDGE
	
	Krok I: vytvoření cyklů
	● EDGE je seznam hran grafu; utřídíme podle opačného lexikografického uspořádání (nejprv 2. složka, potom 1.)
	● SUCCESSOR := EDGE
	● SUCCESSOR utřídíme podle lexikografického uspořádání; při třídění si udržujeme P[h] – původní poloha h-té hrany (i, j) v poli EDGE spolu s hranou (i, j)
	● EDGE a SUCCESSOR definují cykly EDGE[P[h]] = SUCCESSOR[h]
	● čas $O(log \|E\|) = O(log \|V\|) = O(log\ n)$ s $O(m)$ procesory pro n = O(|V|) a m = O(|E|)
	
	Krok II: počítání reprezentantů cyklů
	● reprezentant cyklu = lexikograficky nejmenší hrana cyklu – zdvojováním přes P[h]
	● čas $O(log\ n)$ s $O(m)$ procesory
	● ozn. C množinu reprezentantů (hran) cyklů
	● skonstruujeme bipartitní graf G' = (V', E'), kde V' = V ∪ C, E' = {{u, v} | u ∈ V, v ∈ C, u je v cyklu reprezentovaném v}
	for all 1 ≤ i ≤ m in parallel do
	for EDGE[i] = (u, v) do
		$O(log^2\ n)$	n^2			

EDGE'[2i − 1] := (u, CYCREP[i])
EDGE'[2i] := (v, CYCREP[i])

- čas $O(1)$, $O(m)$ procesorů
- každý vrchol přispěje do EDGE' aspoň dvakrát a může ležet ve více cyklech
- odstraníme duplikáty – lexikograficky utřídíme EDGE a vynecháme duplikáty (kompresí pole)
- přitom si pro každou hranu (u, v) ∈ EDGE' pamatujeme reprezentanta – CERTIFICATE[(u, e)] – hrana vcházející do u a ležící v cyklu obsahujícím hranu e
- celkem m procesorů a čas $O(log\ n)$

Krok III: konstra grafu G'

- kostra T – čas $O(log^2 n)$ s $O(\frac{n^2}{log^2 n})$ procesory
- T je strom, ke každé hraně přidáme opačně orientovanou hranu⇒graf T'
- na T' postavíme Eulerovský cyklus

Krok IV: vytvoření velkého cyklu
Velký cyklus bude obsahovat hrany z T' a z G. Pořadí hran z G definuje Euler. cyklus pro G a hrany z T' tvoří Euler. cyklus pro T'

- definujme cyklické uspořádání hran incidentních s vrcholem w ∈ C
  - {v0, w}, {v1, w}, ..., {vd−1, w}:
- pro 0 ≤ α ≤ d − 1 nechť
  - (i_α, v_α) označuje CERTIFICATE[(v_α, w)] a
  - (v_α, j_α) označuje SUCCESSOR[(i_α, v_α)] po Kroku 1
- upravíme EDGE a SUCCESSOR:
  - do EDGE přidáme hrany z T'
  - ∀w ∈ C, 0 ≤ α ≤ d − 1:
    - SUCCESSOR[(v_α, w)] := (v_α, j_α)
    - SUCCESSOR[(i_α, v_α)] := (w, v_α)
  - $\forall v_\alpha, (w, v^\alpha) \in T'$, nechť v_α susedí s $w_0, w_1, \ldots, w_{d-1} \in C$, potom SUCCESSOR[$(w_i, v_\alpha)$] := $(v_\alpha, w_{i+1 (\mathrm{mod}\ d)})$
  -
  -
- dostávame Eulerův cyklus procházející všemi hranami G ∪ T':
  - Každé w ∈ C definuje procházení jedním cyklem
  - v ∈ V jsou "mosty" mezi cykly z C
  - když "stáhneme" všechny hrany těchto mostů (tj. vynecháme hrany z T'), tak dostaneme Eulerův cyklus pro G
  - složitost $O(1)$ kroků s $O(m)$ procesory