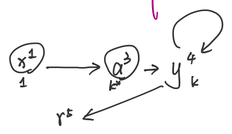
V.S. Longest Path

Tuesday, July 19, 2022 5:50 PM

⇒ while without ctl dep. no recursive dep. ✓

④ ✓

```
x = q(0)y
while k > 0
    a = q(x)
    y = a :: y
r = Σ y
```

↝ 

Definition is good.

↝ Adap = 2.

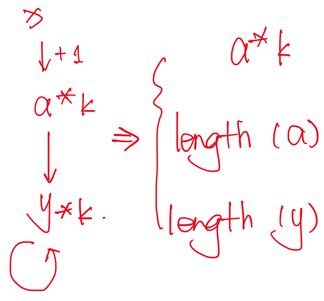
⇒ Naive = 1 + k

↝ refined = 2.

⇒

```
y = []
i = k
x = []
while i > 0 :
    a ← x :: 0
    y ← a :: y
```

↝

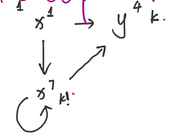


⇒ $\begin{cases} a*k \\ \text{length}(a) = 1*k = k \\ \text{length}(y) = \text{leng}(a)*k = k^2 \end{cases} \Sigma = k+k^2$ V.S. $1+k$

⇒ while without ctl dep. recursive val dep ✓.

⑤

```
while k > 0
    i = k.
    k = k-1
    y = q(x)
    while i > 0 :
        i = i-1
        x = q(x)
```



↝ Adap . = 2 + k!

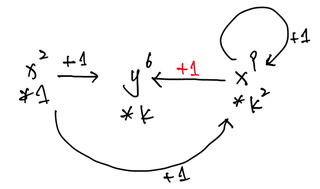
↝ Naive 1 + k + k!

↝ Refined : 2 + k!

⇒

```
[i ← k]^1; [x ← []]^2;
while [i > 0]^3 do
    [i ← i-1]^4; [j ← k]^5;
    [y ← 0 :: x]^6
    while [j > 0]^7 do
        [j ← j-1]^8;
        [x ← 0 :: x]^9
```

⇒



⇒ by longest path single start ⇒ single dest.

length $(x^9) = k^2$.

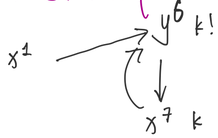
length $(y^6) = k * \text{len}(x^9) = k^3$.

⇒ $\Sigma = k^2 + k^3$. V.S. $k^3 + 1$.

⇒ nested while without ctl dep. recursive val dep ✓

⑥

```
while k > 0
    i = k
    k = k-1
    while i ≥ 0 :
        i = i-1
        [y = q(x)+1]^6
    [x = q(y)]^7
```



↝ Aprog = 2k+1 : ✓.

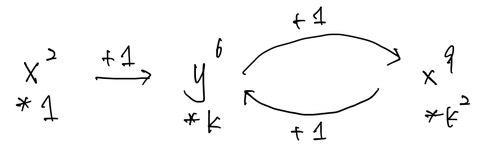
↝ ~~A~~ refined : 2k+1.

↝ Aprog_naive = 1 + k + k! ✓

⇒

```
[i ← k]^1; [x ← []]^2;
while [i > 0]^3 do
    [i ← i-1]^4; [j ← k]^5;
    [y ← 0 :: x]^6
    while [j > 0]^7 do
        [j ← j-1]^8;
        [x ← 0 :: y]^9
```

⇒

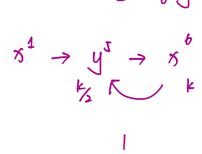


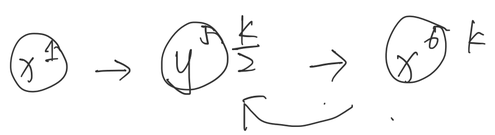
⇒ By longest Path: $\text{len}(x^9) = k + k^2 + 1$, $\text{len}(y^6) = 1 + k + k^2$, $\Sigma = (2 + k + k^2) \times 2$

V.S. $(1 + 2k)$

⑦

```
input k:
[x ← q(0)]^1;
while [k > 0]^2:
    [k = k-1]^3;
    if [k % 2 == 0]^4:
        [y = q(x)]^5;
    [x = q(y)]^6;
```





↝ Aprog = $2 + \frac{k}{2} + k$ ✓.

good for multi path dependency.

↝ Alg_naive = $1 + \frac{k}{2} + k$ ✓

↝ refined = 1 + k. ✓

⇒ while without ctl dep.

(different weighted) multi-path val-dep

the 4 examples above can all be generalized to the implicit rewrite