- 1ワードがnビットの時、aを2進表記するのに必要なワード長 → $\left\lceil \frac{\lg(a+1)}{n} \right\rceil$. =: len(a)

  aの長さ .

- + len(a) その4のを格納するワード → 1.

- 計 . 1 + len(a) ワード を用いる.

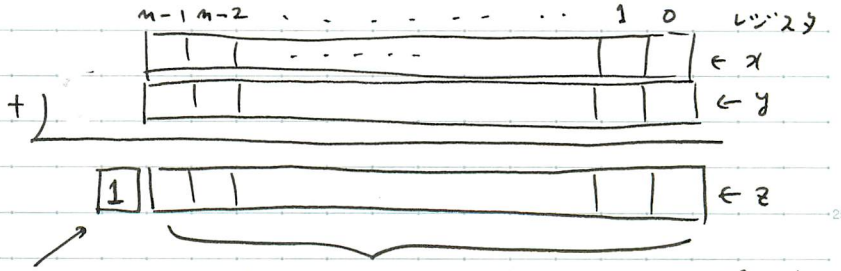
- 問題2.5 : $a, b \in \mathbb{N}$ $n \geq 1$. a+b と ab のビット数の見積もり → 計算結果の格納に用いる.

5/11 ↓

5/15 ↓

⑧ 多倍長整数の加算

- 用いる命題 : 1ワードの整数（2進数）2つの加算.
- 桁あふれ（繰り上がり）の検出 → 補助演算装置.



桁あふれ（繰り上がり）の有無を $\gamma \in \{0,1\}$ とし、値を補助演算装置に格納.

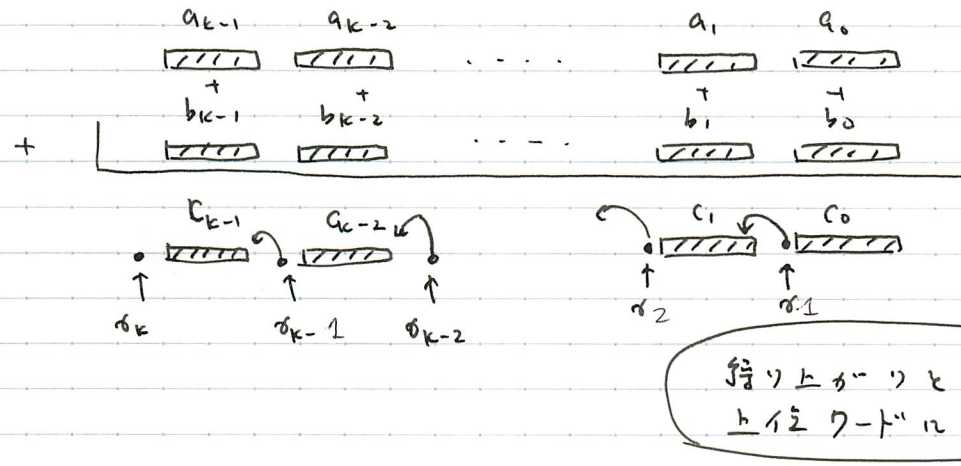
和の下位nビットを z に格納.（桁）

このとき、これとで表す.

$$x + y = \gamma \cdot 2^n + z.$$

$$[z\ \gamma] \leftarrow x + y$$

長さ
- kワードの多倍長整数 a と b の加算.



持ち上がりと次々と上位ワードに加える.

これを数式で表すと.

入力 $\begin{cases} a = [a_0\ a_1\ \cdots\ a_{k-1}] = \sum_{i=0}^{k-1} a_i \cdot 2^{ni}, \\ b = [b_0\ b_1\ \cdots\ b_{k-1}] = \sum_{i=0}^{k-1} b_i \cdot 2^{ni} \end{cases}$

より出力 $c = [c_0\ c_1\ \cdots\ c_{k-1}] = \sum_{i=0}^{k-1} c_i \cdot 2^{ni}$

を得る.

---

⑩ アルゴリズム (多倍長整数の加算)

入力: 多倍長整数 $a = [a_0, a_1 \cdots a_{k-1}]$, $b = [b_0\ b_1 \cdots b_{k-1}]$
出力: 〃 $c = [c_0\ c_1 \cdots c_{k-1}]$ s.t. $c = a+b$.

(0個以上の入力と1個以上の出力をもつ)

---

⑪ アルゴリズム (算法, algorithm)

ある入力(式や値)に対し、ある出力(式や値)を生成するための計算手順(ステップ).

- アルゴリズムの特徴(要件) (Knuth)
  - 停止性: 有限回のステップで停止すること.
  - 正確性: すべてのステップにおける操作が厳密に定められていること.
  - 入力: 0個以上の入力(値)をもつこと.
  - 出力: 1個以上の出力(値)をもつこと.

NO. 15

DATE 4 · 16 · 2015

・有効性：有効な（意味のある）計算が行われること．
（（例）有効でない例：
整数上の Euclid の互除法は，有理数や無限桁の小数を与えれば停止すること．）

◎ アルゴリズム（多倍長整数の加算）

入力：多倍長整数 $a = [a_0\ a_1\ \cdots\ a_{k-1}]$，$b = [b_0\ b_1\ \cdots\ b_{k-1}]$
出力：〃 $c = [c_0\ c_1\ \cdots\ c_{k-1}]$ s.t. $c = a + b$．

(1) $\gamma_0 \leftarrow 0$;
(2) for $i \in [0..k-1]$ do
$[c_i\ \gamma_{i+1}] \leftarrow a_i + b_i + \gamma_i$;
(3) return $[c_0\ c_1\ \cdots\ c_{k-1}]$;

5/18 ↓

←— ここで 計算量 を導入（p.3-1）

◎ 多倍長整数の加算の計算量

☆ 計算量のワードごとの四則演算の回数を単位とする（p.4）

上のアルゴリズムの計算量（演算回数）を調べると…．

(1) $\gamma_0 \leftarrow 0$; 代入は今回は無視
(2) for $i \in [0..k-1]$ do ループ k回
$[c_i\ \gamma_{i+1}] \leftarrow a_i + b_i + \gamma_i$; 加算 2回
(3) return $[c_0\ c_1\ \cdots\ c_{k-1}]$; return 文は無視

より，全体で行われる演算は 加算 $2k$ 回．
よって，このアルゴリズムの計算量は $2k = O(k)$
$= \Theta(k)$．

すなわち，多倍長整数の加算の計算量は，整数の長さ（ワード長）k に比例する．

5/18 cont'd.

計算量 Computational complexity

算法の効率を測る上での理論的指標の一つ.

この授業で扱う
- ◎ 時間計算量 (time complexity) : 必要な計算のステップ数. 5/18
- ○ 空間計算量 (space complexity) : 計算に必要な記憶容量.

◎ 計算量の表し方 … 「漸近記法」 (§ 1.2) 5/25

$$T : \mathbb{N} = \{0,1,2,\dots\} \longrightarrow \mathbb{R}_{\geq 0}$$
$$\Psi \qquad\qquad\qquad \Psi$$
$$n \qquad \mapsto \qquad T(n)$$
$$\uparrow$$

入力データの「サイズ」 … 数の桁数, ワード数, 多項式の次数, ...

(例) $f(n)$ n: 多倍長整数のワード数.

$f(n) = an$, $g(n) = bn^2$, $a>0$, $b>0$

⟶ いずれも g が f と「追いぬく」

⟶ f の方が, 計算量が「漸近的に」よりよい.

他の例: $h(n) = c^n$ $(c>0)$, $p(n) = d\log(n)$ $(d>0)$ 等. ◎

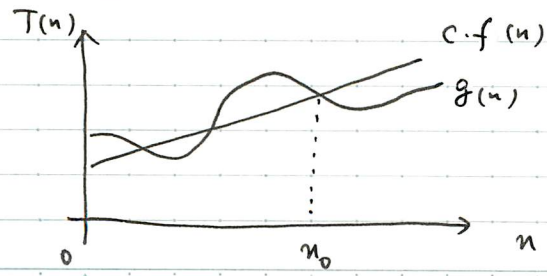
◎ 計算量の記法 (notations)

Def 1.1 (漸近記法). $f(n), g(n) : \mathbb{N} \longrightarrow \mathbb{R}$

① O-notation : 漸近的上界.

$$g(n) = O(f(n)) \overset{\text{def}}{\Leftrightarrow} \exists c>0,\ \exists n_0>0 \text{ s.t. } n \geq n_0 \Rightarrow 0 \leq g(n) \leq c\cdot f(n)$$

テキストで $\in$ となっているのと本質的に変わらない. 以下同様.



(数学の O 表記と同じ)