**Hypercube problems**

# Lecture 18

*December 14, 2017*

Lecturer: Petr Gregor | Scribe by: Martin Dvorak | Updated: August 16, 2018

## 1 Definition of binary counters

**Motivation** How to efficiently represent integers in memory if we need only increment and decrement?

**Definition 1** *By a space-optimal (non-redundant) representation of integers* $\{0, \ldots, 2^n - 1\}$ *we mean any bijection* $r : \{0, \ldots, 2^n - 1\} \to \{0, 1\}^n$.

**Example** $\mathrm{SBC}(a) = w_n \ldots w_1$ such that $a = \sum_{i=1}^{n} w_i 2^{i-1}$ (a standard binary code)

**Example** $\mathrm{SRGC}(a)$ = a-th vertex in the standard reflected Gray code $\Gamma_n$

$$\Gamma_1 = (0, 1)$$

$$\Gamma_{n+1} = (0\Gamma_n, 1\Gamma_n^R)$$

**Definition 2** *Bitprobe model - how many bit reads/writes in the data structure are needed in the worst/average case for each operation? The overhead of operations to determine which bits to read/write is omitted.*

**Example**

| | space | read | write | average read | average write |
|---|---|---|---|---|---|
| SBC | $n$ | $n$ | $n$ | $2 - 2^{1-n}$ | $2 - 2^{1-n}$ |
| SRGC | $n$ | $n$ | 1 | $n$ | 1 |

**Definition 3** *(non-redundant counters): An* $(n, r, w)$*-counter is a data structure that uses* $n$ *bits to represent integers* $\{0, \ldots, 2^n - 1\}$ *with increment and decrement operations (modulo* $2^n$*) with* $r$ *bits read and* $w$ *bits written for each operation in the worst case (in the bitprobe model).*

**Definition 4** *A decision assignment tree (DAT) is a binary tree to represent increment/decrement operations. Inner vertices are labeled by bits that are read, left/right subtrees represent the cases when the read bit is 0/1, leaves* $l$ *are labeled by sets* $W_l$ *of assignments of type* $x_i := b, b \in \{0, 1\}$.

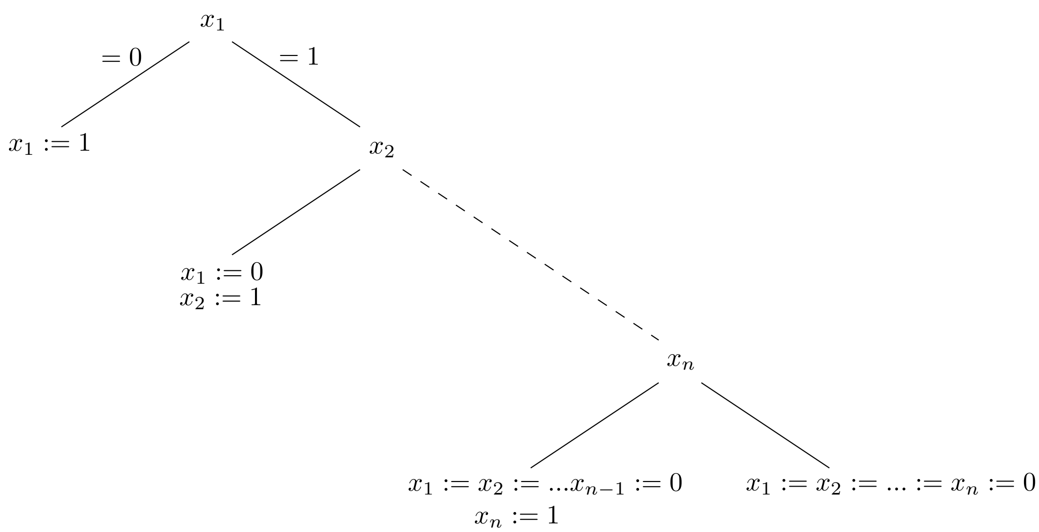


**Figure 1**: DAT for SBC.

**Example** DAT for increment in SBC is on Figure 1. DAT determines read/write complexity as follows:
(worst case) write $= max_{l:leaf}|W_l|$
(worst case) read = depth of DAT
average read = weighted average depth (expected depth)

**Example** Increment in SRGC is easily transformed to DAT.

$$inc(x_n, \ldots, x_1) = \begin{cases} x \oplus e_1 & \text{if } x_n \oplus \cdots \oplus x_1 = 0 \\ x \oplus e_{min\{i+1,n\}} & \text{else where } i = \text{the smallest such that } x_i = 1 \end{cases}$$

**Definition 5** *(redundant counters): An $(n, e, r, w)$-counter with efficiency $e = \frac{L}{2^n}$ is a data structure to represent integer $\{0, \ldots, L-1\}$ for increment/decrement (modulo $L$) with worst case read/write complexity $r/w$ (in the bitprobe model).*

## 2 Basic properties of binary counters

**Question** How many bit reads are needed?

**Observation 6** *In any space optimal counter, all written bits (in leaves) have been read.*

**Proposition 7 ([6])** *Any $(n, r, w)$-counter requires at least $r \geq \log_2 n + 1$ reads for increment (or decrement).*

**Proof** Suppose there is an $(n, r, w)$-counter with $r \leq \log_2 n$. Consider corresponding DAT $T$ for increment (or decrement). $T$ has at most $2^r - 1 \leq n - 1$ internal vertices. Thus some bit is never read $\implies$ never written (by Observation) $\implies$ contradiction with space equality. ■

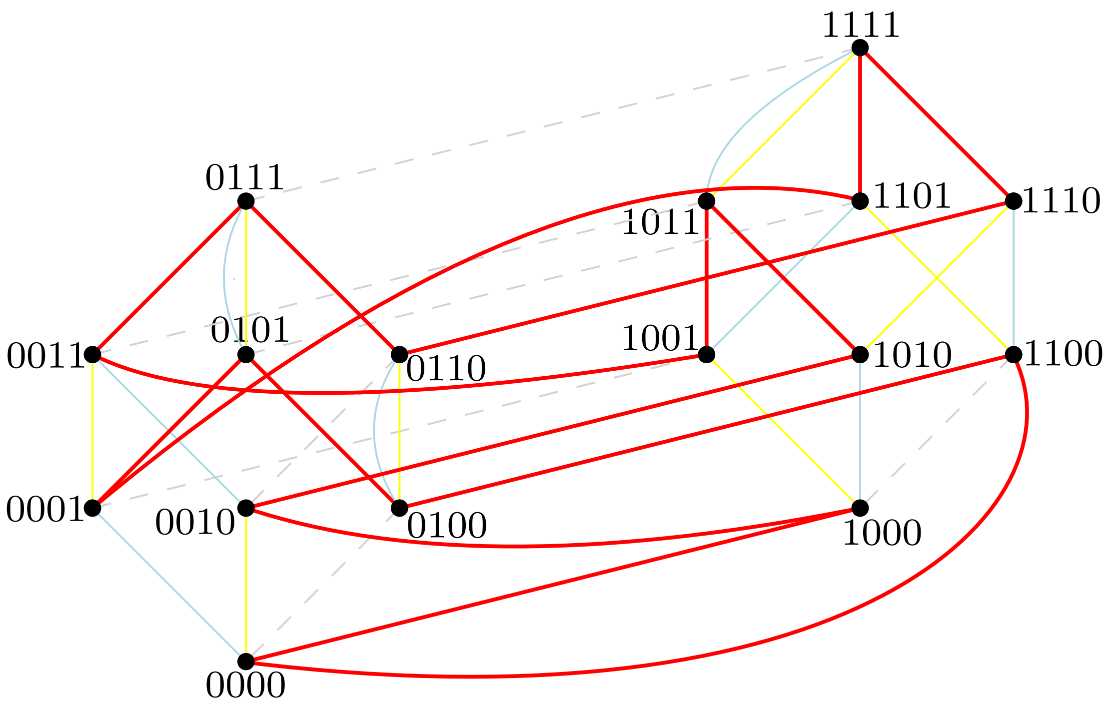


**Figure 2**: A quasi-Gray code for $Q_4$: bold red lines are increment/decrement edges, pale yellow lines denote same direction of increment, pale blue lines denote the same direction of decrement and dashed gray lines are ordinary edges of the hypercube.

**Proposition 8 ([3])** *Any $(n, \frac{L}{2^n}, r, 1)$-counter requires at least $r \geq \log_2 \log_2 L + 1$ reads for increment (or decrement).*

**Proof** To represent $L$ integers, at least $\log_2 L$ bits need to be modified, at least once set to 0, once to 1. The corresponding DAT has at least $2 \log_2 L$ leaves. Since its depth is $r$, it has at most $2^r$ leaves, so $2^r \geq 2 \log_2 L$. ■

## 3 Binary counters with small worst-case reads

**Question** Does every $(n, r, 1)$-counter require $r = n$ reads? No!

**Proposition 9 ([2])** *There is a $(4, 3, 2)$-counter for both increment/decrement.*

**Proof** Consider the following quasi-Gray code (with distance $\leq 2$ of consecutive strings) on Figure 2. Vertices can be recursively cut into $Q_2$'s (pairs of adjacent vertices) so that they have the same incoming and outgoing directions. This gives DATs for increment and decrement, see Figures 3 and 4. ■

**Question** Can we improve it from $w = 2$ to $w = 1$?

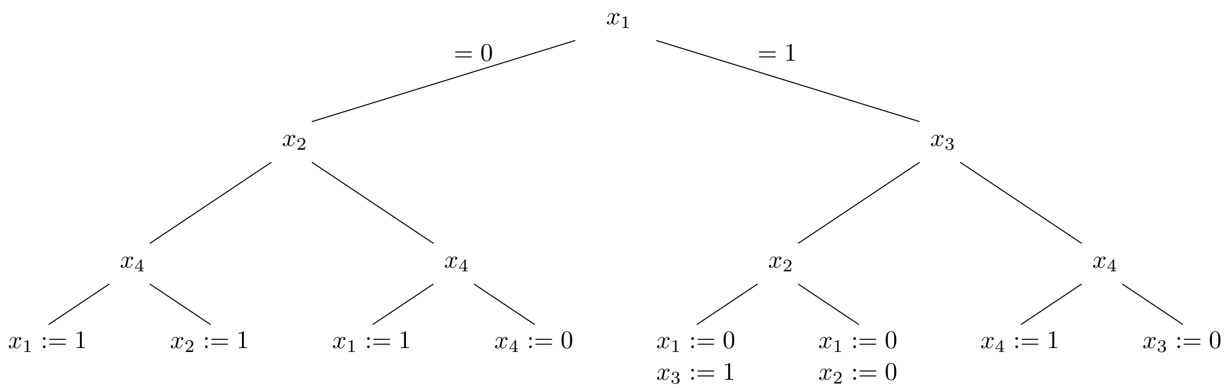


**Figure 3**: DAT for increment in the $(4, 3, 2)$-counter.

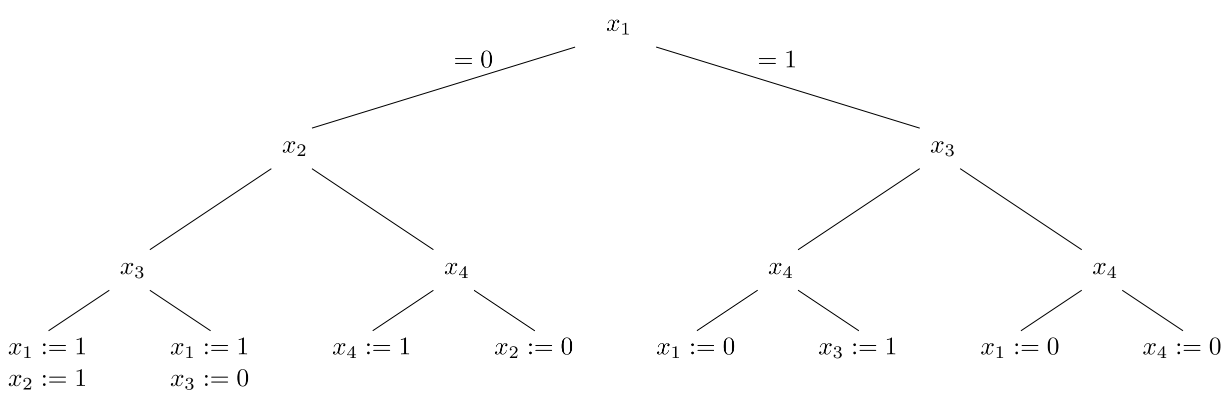


**Figure 4**: DAT for decrement in the $(4, 3, 2)$-counter.

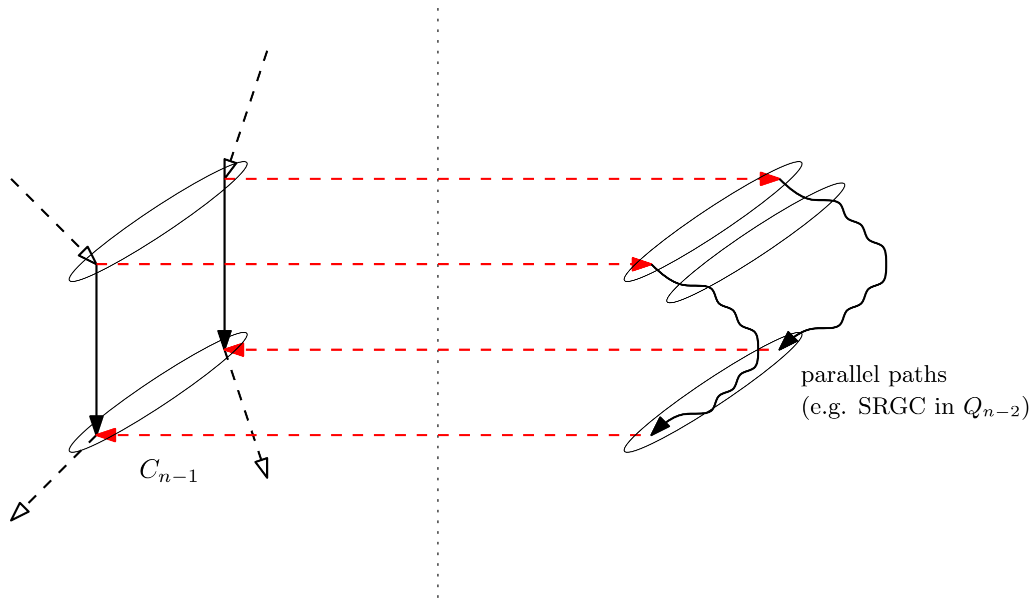


**Figure 5**: Induction step in $Q_n$ for $(n, n-1, 1)$-counter.

**Theorem 10 ([4])** *There is a $(5, 4, 1)$-counter for both increment and decrement.*

**Proof** Modify the previous quasi-Gray code in $Q_4$ onto Gray code in $Q_5$ by replacing the distance 2 steps with path-partition through the other copy of $Q_4$. The recursive cutting into adjacent with the same outgoing / incoming edges can be still found. ■

**Corollary 11 ([4])** *There is an $(n, n-1, 1)$-counter for both increment and decrement for every $n \geq 5$.*

**Proof** By induction on $n$, see Figure 5. In the counter $C_{n-1}$ for $Q_{n-1}$, delete one pair of parallel edges and interleave the circle by two parallel paths in two $Q_{n-2}$ on the other $Q_{n-1}$. Note that the resulting Gray code supports both increment and decrement in $n-1$ reads. ■

**Problem** ([4]) Does an $(n, n-2, c)$-counter exist (for some constant $c$ and sufficiently large $n$)? For example $(6, 4, c)$ is the first open case.

**Note 12** *There exists an $(n, \frac{1}{2}, \log_2 n + 2, 3)$-counter for increment (no decrement) [2].*

**Note 13** *There is a ternary $(n, O(\log n), 2)$-counter (in general for every odd-sized alphabet) [5].*

## 4 Binary counters with small average-case reads

**Proposition 14 ([4])** *Any (n,r,w)-counter needs at least $2 - 2^{1-n}$ bit reads on average (thus SBC is optimal in this respect).*

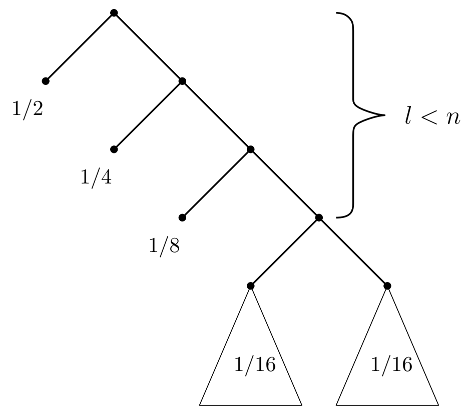


**Figure 6**: Lower bound for average bit reads.

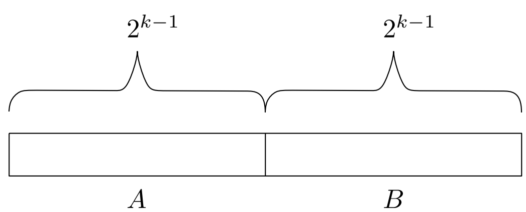


**Figure 7**: Halves of Recursive partition Gray code.

**Proof** DAT needs to have the structure of DAT of SBC (up to isomorphism), where every inner vertex has a leaf as one of its children. Otherwise let $l$ denote the smallest level where a vertex does not have a leaf as one of its children. Average read count is then at least:

$$\sum_{i=1}^{l-1}(i \cdot 2^{-i}) + l \cdot 2^{1-l} = 2 - 2^{1-l} > 2 - 2^{1-n}$$

See Figure 6 for an illustration. ■

**Theorem 15 ([1])** *There is a an $(n, n, 1)$-counter with at most $4 \log n$ reads on average if $n = 2^k$ for some positive integer $k$.*

**Proof** Consider the following recursive partition Gray code:

$$RPGC_1 = (0, 1)$$

For the step $RPGC_{2^{k-1}} \to RPGC_{2^k}$ see Figure 7. Let $A, B$ be the first and second $2^{k-1}$ bits of the code word $(A, B)$ in $RPGC_{2^k}$. Then increment and decrement is as follows.

$$inc(A, B) = \begin{cases} (A, dec(B)) & \text{if } A = B \\ (inc(A), B) & \text{else} \end{cases}$$

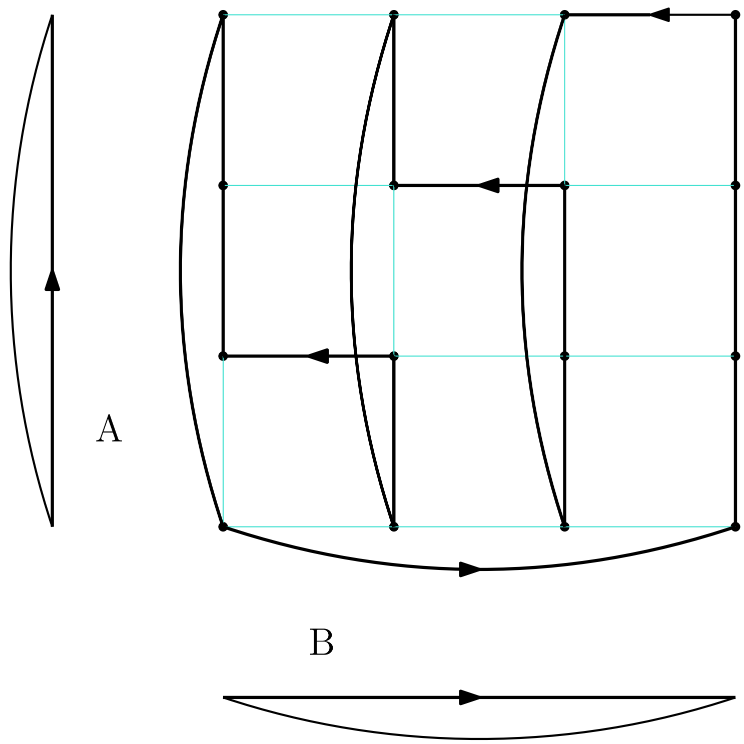


**Figure 8**: A construction of Recursive partition Gray code.

The increment operation is explained in Figure 8.

$$dec(A,B) = \begin{cases} (A, inc(B)) & \text{if } A = succ(B) \\ (dec(A), B) & \text{else} \end{cases}$$

What is the average read count? Take $(A,B)$ at random. Let $r_{inc}(A,B)$ resp. $r_{dec}(A,B)$ denote the count of reads for increment resp. decrement operation on $(A,B)$. Let $c(A,B)$ denote the average count of reads to compare whether $A = B$. Now $|A| = |B| = n$.

We need to prove that $r_{inc}(A,B) \leq 4\log n$ and also $r_{dec}(A,B) \leq 4\log n$. We will prove both inequalities at the same time using mathematical induction.

If $n = 2$, then we perform $r = 2$ reads, which is less than $4\log 2 = 4$.

Let's do the induction step from $n$ to $2n$ now. If we calculate the probabilities of reading every bit during the compare operation, we get:

$$\mathbb{E}[c(A,B)] = 2 \cdot \sum_{i=0}^{n-1}(2^{-i}) = 2 \cdot (2 - 2 \cdot 2^{-n}) = 4 - 2^{2-n}$$

The average read count for increment is as follows. Decrement is analogous.

$$\mathbb{E}[r_{inc}(A,B)] =$$

$$= \mathbb{E}[c(A,B)] + \mathbb{E}[r_{dec}(B)|A = B] + \mathbb{E}[r_{inc}(A)|A \neq B] =$$

$$= 4 - 2^{2-n} + \mathbb{E}[r_{dec}(B)] \cdot 2^{-n} + \mathbb{E}[r_{inc}(A)] \cdot (1 - 2^{-n}) =$$
$$= 4 - 2^{2-n} + 4 \log n \cdot 2^{-n} + 4 \log n \cdot (1 - 2^{-n}) =$$
$$= 4 - 2^{2-n} + 4 \log n \leq$$
$$\leq 4 \log n + 4 = 4 \log(2n)$$

■

**Note 16** *For general $n$, there is an $(n, n, 1)$-counter with at most $6 \log n$ reads on average [1].*

**Problem** ([4]) For general $n$, is there any $(n, n-1, 1)$-counter with $O(\log n)$ average reads?

## Notes

A short survey of binary counters with additional references can be found in [4]. This lecture does not cover most recent results in [5, 7].

## References


[1] Prosenjit Bose, Paz Carmi, Dana Jansens, Anil Maheshwari, Pat Morin, and Michiel Smid, *Improved methods for generating quasi-gray codes*, In Proceedings of the 12th Scandinavian Conference on Algorithm Theory, SWAT '10, pages 224–235, Springer-Verlag, 2010.

[2] Gerth Stølting Brodal, Mark Greve, Vineet Pandey, and Srinivasa Rao Satti, *Integer representations towards effcient counting in the bit probe model*, Journal of Discrete Algorithms, 26:34–44, 2014.

[3] Michael L. Fredman, *Observations on the complexity of generating quasi-gray codes*, SIAM Journal on Computing, 7(2):134–146, 1978.

[4] Zachary Frenette, *Towards the efficient generation of Gray codes in the bitprobe model*, Master's thesis, University of Waterloo, Waterloo, Ontario, Canada, 2016.

[5] Diptarka Chakraborty, Debarati Das, Michal Koucký, and Nitin Saurabh, *Optimal Quasi-Gray Codes: Does the Alphabet Matter?*, In Proceedings of 26th Annual European Symposium on Algorithms (ESA 2018), Leibniz International Proceedings in Informatics (LIPIcs), vol. 112, pages 12:1–12:15, 2018.

[6] Patrick K. Nicholson, Venkatesh Raman, and S. Srinivasa Rao, *A survey of data structures in the bitprobe model*, In Space-Efficient Data Structures, Streams, and Algorithms, volume 8066, pages 303–318, Springer Berlin Heidelberg, 2013.

[7] Mikhail Raskin, *A linear Lower Bound for Incrementing a Space-Optimal Integer Representation in the Bit-Probe Model*, In Proceedings of 44th International Colloquium on Automata, Languages, and Programming (ICALP 2017), Leibniz International Proceedings in Informatics (LIPIcs), vol. 80, pages 88:1–88:12, 2017.